Mining Coherent Subgraphs in Multi-Layer Graphs with Edge Labels

Brigitte Boden, Stephan Günnemann, Holger Hoffmann, and Thomas Seidl
Data Management and Data Exploration Group
RWTH Aachen University, Germany
{boden, guennemann, h.hoffmann, seidl}@cs.rwth-aachen.de

ABSTRACT

Mining dense subgraphs such as cliques or quasi-cliques is an important graph mining problem and closely related to the notion of graph clustering. In various applications, graphs are enriched by additional information. For example, we can observe graphs representing different types of relations between the vertices. These multiple edge types can also be viewed as different "layers" of the same graph, which is denoted as a "multi-layer graph" in this work. Additionally, each edge might be annotated by a label characterizing the given relation in more detail. By exploiting all these different kinds of information, the detection of more interesting clusters in the graph can be supported.

In this work, we introduce the multi-layer coherent subgraph (MLCS) model, which defines clusters of vertices that are *densely connected* by edges with *similar labels* in a *subset* of the graph layers. We avoid redundancy in the result by selecting only the most interesting, non-redundant clusters for the output. Based on this model, we introduce the best-first search algorithm MiMAG. In thorough experiments we demonstrate the strengths of MiMAG in comparison with related approaches on synthetic as well as real-world datasets.

Categories and Subject Descriptors: H.2.8 Database management: Database applications [Data mining]

Keywords: dense subgraphs, graph clustering, networks

1. INTRODUCTION

Mining graph and network data has gained much attention in recent years. One important mining task is *graph clustering* that aims at grouping the vertices of a graph into so-called *clusters* such that many edges between vertices of the same cluster exist, i.e. the vertices are *densely connected*. This task is often also referred to as "dense subgraph mining" including the detection of cliques or quasi-cliques [12].

Besides the mere graph data, real-world data often contains additional information, which can be exploited by clustering approaches. Several approaches have been proposed considering additional information about the *vertices* of a graph (e.g. [19, 14, 7, 6]). In this work, however, we consider additional information about the *edges* of a graph.

For example, a graph can contain *different types of edges*, which represent different types of relations between vertices. Such different types can occur, for example, when we combine information from several information networks: e.g., combining a co-author network with a citation network. In the first graph, authors are connected if they have common papers; in the second graph, if a paper of one author cited a paper of the other author. Thus, we will get two edge types: "co-authorship" and "citation". Furthermore, each edge might also be associated with a label, e.g. the number of co-authored (or cited) papers. We denote such a graph with multiple edge types as a "multi-layer graph". It is defined as a set of graphs (called "layers") where each graph is based on the same set of vertices and represents the edges of one certain type. Accordingly, in each layer a different edge set is given. These layers can also be viewed as "dimensions" of the graph. (In the following, we use the terms "layers" and "dimensions" interchangeably.) An exemplary multi-layer graph is depicted in Fig. 1; several real-world examples are described in the experimental section.

For graphs with edge labels, we can distinguish between two possible interpretations of the labels: First, labels can be regarded as edge *weights* that denote the strength of the relation between the incident vertices. In this paper, however, we consider a second interpretation: the edge labels represent *characteristics* of the relations. For example, a co-author network might contain information about the collaboration between two authors, as the begin or end time of the collaboration, research topics, conferences/journals where the joint papers were published etc.

Overall, in this work, we aim at finding clusters of vertices that are *densely connected* by edges with *similar* labels in a *subset* of the graph layers. These clusters are denoted as (multi-layer) *coherent subgraphs*.

We want to highlight that the coherent subgraphs need not to appear across all layers, but we detect them in subsets of the layers. This is important as some of the edge types might not be relevant for finding interesting coherent subgraphs at all; other types might be relevant only for certain subgraphs. Thus, for each coherent subgraph we find an *individual* set of relevant layers. This principle is motivated by the field of *subspace clustering* [10] that aims at analyzing subsets of the dimensions in a vector space and that stems from the fact that in higher-dimensional vector data it is unlikely to find objects that are similar w.r.t. all

KDD'12, August 12–16, 2012, Beijing, China.

of their characteristics; each cluster is associated with an individual subspace projection. Exploiting these observations in our model, the detected coherent subgraphs are useful in several scenarios: We might find a dense cluster of authors who started their collaboration at a similar time and co-authored papers at the same conferences. The authors in another dense cluster might have cited each others' papers on similar research topics, but not have published joint papers. Similar, in a co-actor graph representing the joint work of actors (cf. Sec. 5), a cluster might be a group of actors who worked together on movies with similar success.

Additionally, we consider the fact that a vertex can naturally belong to more than one cluster, e.g. an author might of course participate in several working groups. Thus, we allow our clusters to overlap. However, similar to subspace clustering, allowing overlap can lead to a huge number of valid clusters that mostly represent redundant information [13, 7]. Thus, we propose a clustering model that allows clusters to overlap to a certain extend, but avoids redundancy in the resulting set of clusters. This final set of clusters ("clustering") should contain the *most interesting clusters* w.r.t. a quality function which can be specified by the user.

Finally, since determining the overall clustering according to our model is NP-hard (as also with most dense subgraph mining models on a single graph), we introduce the algorithm MiMAG using a best-first search [16] to find an approximate solution. Best-first search is an established search principle to explore a graph, which in our case is a search tree for enumerating subgraphs, in an informed fashion. Starting in an initial node (in our case: the root node of the search tree), best-first search algorithms iteratively expand the "most promising" node based on a given heuristic. In MiMAG, the most promising subgraphs are expanded to detect the most interesting clusters first. This concept is related to the well-known A* algorithm for finding minimum-cost paths in a graph [9].

The main contributions of this paper are the following:

1. We propose the new paradigm of clustering multi-layer graphs with edge labels.
2. We introduce the clustering model MLCS, which avoids redundancy in the result set.
3. We propose the best-first search algorithm MiMAG to approximate the MLCS clustering.

2. RELATED WORK

For mining graph data there exist various mining tasks [1]. Some of the most active areas are graph clustering, graph classification, and frequent subgraph mining. In this work, we concentrate on *clustering graph/network data*. An overview of the various existing models and techniques is given in [1] and [4]. The term "graph clustering" is somewhat ambiguous as it is (a) used for the clustering of graph databases, where a cluster represents a *set of graphs*, and (b) for clustering in one large graph, where the clusters represent *sets of vertices* from the graph. The latter is often also referred to as "dense subgraph mining", and is the meaning that is used in this paper. For the definition of dense subgraphs, several models exist. Two of the most widely-used models are cliques and γ-quasi-cliques [17].

The concept of subspace clustering was developed for the task of clustering vector data. Traditionally, clustering is done by using all dimensions of the feature space. However, full-space clustering does not scale to high-dimensional data since locally irrelevant dimensions may obfuscate the clustering structure [2, 10]. As a solution, *subspace clustering* methods detect an individual set of relevant dimensions for each cluster [10]. For subspace clustering, several models and algorithms have been proposed. E.g., cell-based subspace clustering methods obtain high-quality results and are efficiently computable [15].

Some clustering approaches have been proposed that consider graphs with labeled *vertices* (here, the vertex labels are vectors). These approaches can be seen as a combination of graph clustering with clustering approaches for vector data. However, they mostly rely on fullspace-clustering on the vertex attributes (e.g. [19]). Recently, the approaches [14, 7, 6] were introduced to deal with the combination of subspace clustering and dense subgraph mining. In these approaches, clusters are ensured to be densely connected and vertices in a cluster are similar in subsets of their attribute values. Although these cluster models are related to the model introduced in this paper, adapting these methods to handle *edge* labels does not lead to the desired results. Two approaches for such an adaption are discussed and evaluated in the experimental section.

Graphs with a *single* edge type and labels in terms of edge *weights* can be considered by some graph clustering approaches like minimum cut [1] and spectral clustering [19].

To the best of our knowledge, there are no previous approaches for clustering in *multi-layer graphs with edge labels.* Also in the survey by Fortunato [4] it is mentioned that such graphs have not been dealt with by any algorithm. Even though the authors of [11] mention the existence of different types of relations in a network (which they call "levels of relation"), they just summarize all the different relations between two vertices into a single edge weight.

A concept which is related to our approach is the detection of cross-graph quasi-cliques [17]. Given a database of graphs each having the same vertices, a cross-graph quasi-clique is defined as a set of vertices that forms a quasi-clique in *all* of the graphs. Only maximal sets having this property are output. The approaches [20] and [21] also work on a graph database and mine sets of vertices that form a clique [20] or quasi-clique [21] in at least a certain percentage of the graphs in the database (which is called the "support" of the (quasi-)clique). Both approaches aim at mining *closed* (quasi-)cliques, i.e. a (quasi-)clique O is not contained in the output if one of O's supersets also forms a (quasi-)clique having the same support. In [3], cross-graph cliques are detected in a dynamic graph represented as a 3-dimensional boolean cube. To adapt these approaches to our problem setting, the graphs in the graph database could be seen as the different layers of our input graph. However, edge labels are not considered by these approaches. Furthermore, the existing methods do not avoid redundancy in the result set apart from simply excluding *subsets* of (quasi-)cliques. Thus their output can often contain a large set of highly overlapping vertex sets. In our experimental section, we compare our approach to an adaption of the closed quasi-clique mining algorithm Cocain [21].

3. MODEL

In this section, we introduce the MLCS ("Multi-layer coherent subgraph") model for the clustering of graphs with different types of edges. We start by providing a formal definition of the input graph. For ease of presentation, we represent the input graph as a *set* of graphs (called "multi-layer graph") each having the same vertex set V; each graph contains the edges of one type with their corresponding labels. Formally, the layer graph is defined as:

Definition 1 (Multi-Layer Graph). *A multi-layer graph $\mathcal{G}$ for a set of dimensions $Dim = \{1, \ldots, d\}$ is a set $\mathcal{G} = \{G_i \mid i \in Dim\}$ of graphs*

$$G_i = (V, E_i, l_i), E_i \subseteq V \times V, l_i : E_i \to \mathbb{R}$$

where each graph layer $G_i, i \in Dim$ is an undirected graph without self-loops and with an edge labeling function l_i.

We can easily handle graphs with different vertex sets V_i by simply considering the union $V = \bigcup V_i$. The remainder of this section is structured as follows: In Section 3.1, we introduce our definition for a single cluster. We define our redundancy model and selection criteria for the final clustering in Section 3.2. In Section 3.3 we introduce the cluster quality function that is used in our experiments.

3.1 Cluster model

As discussed in the introduction, an MLCS cluster is a set of vertices which are connected with a *high density* by edges with *similar labels* in a *subspace of the multi-layer graph* (i.e. in a subset of the graph layers).

Cluster property for a single graph layer. First, we consider a single graph layer G_i. For the density of a subgraph, we use the established *quasi-clique* model [17, 21, 12]. The quasi-clique model defines dense subgraphs based on their intra-cluster connectivity. Formally,

Definition 2 (γ-quasi-clique). *A vertex set $O \subseteq V$ in a graph $G = (V, E)$ is a γ-quasi-clique for $\gamma \in [0, 1]$ if*

$$\forall v \in O : deg_G^O(v) \geq \lceil \gamma \cdot (|O| - 1) \rceil$$

where $deg_G^O(v) = |\{u \in O \mid (u, v) \in E\}|$. The density of a quasi-clique O in graph layer G_i is defined by

$$\gamma_{G_i}(O) = \frac{\min_{v \in O}\{deg_{G_i}^O(v)\}}{|O| - 1}$$

For our cluster model, we consider a vertex set as dense if it is a 0.5-quasi-clique, i.e. its quasi-clique density is at least 0.5. As shown in [21], for $\gamma \geq 0.5$ the vertices in a γ-quasi-clique are connected "tightly and relatively evenly". This also ensures that the subgraph is connected in the graph [21]. For the similarity of the edge labels, we use a cell-based cluster model [15]. To be considered similar, the labels of the edges in a cluster may vary at most by a threshold w. Formally, we define a cluster in a graph layer G_i as follows:

Definition 3 (One-dimensional MLCS cluster). *A vertex set $O \subseteq V$ is a one-dimensional MLCS cluster in a graph layer $G_i = (V, E_i, l_i)$ (w.r.t. threshold w and distance function dist) if it forms a 0.5-quasi-clique in the graph G_i and*

$$\forall x, y \in E_i(O) : dist(l_i(x), l_i(y)) \leq w$$

where the edge set $E_i(O)$ is defined as $E_i(O) = \{(u, v) \in E_i \mid u, v \in O\}$

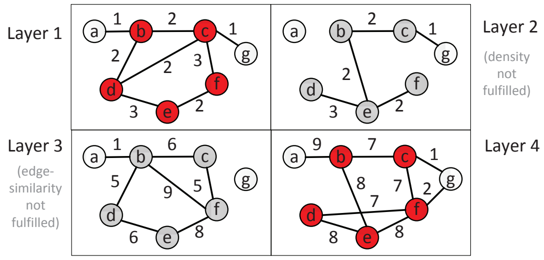


Figure 1: Exemplary MLCS cluster in layer 1 & 4

Please note that the threshold w is only needed if the edge labels are continuous valued. If we have categorical labels or the layer graphs are unlabeled, we can simply set $w = 0$. Additionally, since two nodes connected by a single edge trivially fulfill the definition, we only consider subgraphs with at least 3 vertices as clusters.

Cluster property in subspaces of the multi-layer graph. Next, we consider clusters located in subspaces of the layer graph. Naturally, the vertex set of a multi-dimensional cluster should fulfill the one-dimensional cluster property for all of the dimensions in the subspace. This idea leads to some important observations: If an edge (u, v) exists in one graph layer, it does not automatically exist in another layer. Thus, when we consider the *same vertex set* in different graph layers, the corresponding *edge sets can differ* from each other. Thus, in our model, we ensure the density of the subgraph for each layer individually. Formally,

Definition 4 (MLCS cluster). *An MLCS cluster $C = (O, S)$ in a multi-layer graph $\mathcal{G} = \{G_i \mid i \in Dim\}$ consists of a vertex set $O \subseteq V$ and a non-empty set of relevant layers $S \subseteq Dim$ such that $\forall i \in Dim : i \in S \Leftrightarrow O$ is an MLCS cluster in the graph layer G_i.*

The density of the cluster $C = (O, S)$ is defined as

$$\gamma_S(O) = \tfrac{1}{|S|} \textstyle\sum_{i \in S} \gamma_{G_i}(O)$$

Since the edge sets per layer may differ, also the cluster's density in each layer may vary. Thus, we define the density of the cluster as the average density over all layers in the subspace. Note that in the case of unlabeled layer graphs, our cluster model resembles the definition of cross-graph quasi-cliques [17] or closed quasi-cliques [21].

In Fig. 1, the vertex set $O = \{b, c, d, e, f\}$ forms an MLCS cluster for $w = 1$ in the layers 1 and 4. In layer 2, O is not a quasi-clique and in layer 3, the edge labels of $E_3(O)$ are not similar, thus O does not form a cluster in these layers.

3.2 Clustering model

In the previous section we introduced the properties that a vertex set has to fulfill to form an MLCS cluster. As motivated in the introduction, the clusters are allowed to overlap. However, just outputting all valid clusters can lead to a large amount of valid clusters that are possibly very similar to each other and thus contain redundant information. An example (for simplicity without edge labels) is shown in Fig. 2, where the clusters C_2 and C_3 highly overlap in layer 1. Thus, the final clustering result should be a non-redundant set of the "most interesting" clusters. This result set is called the *MLCS clustering*.

As the "interestingness" of a cluster can be highly application dependent, it is defined by a quality function $Q(C)$,

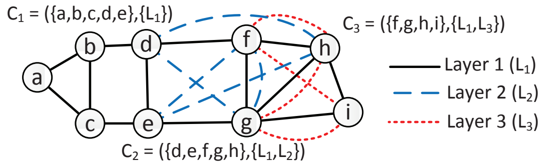


Figure 2: Overlapping clusters with given qualities $\mathbf{Q(C_1) = 2.5}$, $\mathbf{Q(C_2) = 5}$ and $\mathbf{Q(C_3) = 5.3}$

which can be specified by the user. The default quality function that was used in our experiments is introduced in Section 3.3.

For the avoidance of redundancy, we first introduce a redundancy relation. We define a cluster C to be redundant w.r.t. a cluster C' if a significant fraction of C's edges is also covered by C' (thus, they represent similar information) and the quality of C' is not smaller than that of C. Formally,

Definition 5 (Redundancy relation). *A cluster $C = (O, S)$ is redundant w.r.t. a cluster $C' = (O', S')$ (short: $C \prec_{red} C'$) if*

$$C \neq C' \wedge Q(C) \leq Q(C') \wedge \frac{1}{|S|} \sum_{i \in S \cap S'} \frac{|E_i(O) \cap E_i(O')|}{|E_i(O)|} \geq r$$

for the redundancy parameter $r \in (0, 1]$.

In our experiments, $r = 0.25$ proved to be a good choice, thus this is used as the default redundancy parameter. In Fig. 2, the cluster C_2 is redundant w.r.t. the cluster C_3. In contrast, C_1 is not redundant w.r.t. C_2. Although it's quality is lower, the edge overlap between the clusters is below the threshold. Please note that two clusters with equal quality might be pairwise redundant w.r.t. each other.

Based on this redundancy relation we now select the maximum-quality clustering *Result* from the set $\mathcal{A}$ of all valid clusters. This clustering should not contain clusters that are redundant w.r.t. each other and at the same time it should maximize the sum of the qualities of the selected clusters:

Definition 6 (MLCS clustering). *Given a multi-layer graph $\mathcal{G}$ and the set $\mathcal{A}$ of all valid MLCS clusters, the maximum-quality clustering Result $\subseteq \mathcal{A}$ fulfills:*

- *Redundancy-freeness: $\neg\exists C, C' \in Result : C \prec_{red} C'$*
- *Maximum quality sum: $\neg\exists Result' \subseteq \mathcal{A}$: $Result'$ is redundancy-free and $\sum_{C \in Result'} Q(C) > \sum_{C \in Result} Q(C)$.*

In Fig. 2, the clustering solution would be $\{C_1, C_3\}$.

Complexity results. We briefly describe the main result of our complexity analysis:

Theorem 1. *Given a layer graph $\mathcal{G}$ over the vertices V, determining the MLCS clustering is NP-hard w.r.t. $|V|$.*

Proof. We prove this theorem by a polynomial reduction of the NP-hard k-clique problem ("Is there a clique with at least k vertices"?) to the determination of an *MLCS* clustering. Given a graph $G = (V, E)$ and an integer k, we can solve the k-clique problem as follows: Take $\mathcal{G} = \{G_1 = (V, E, l_1)\}$ with $l_1(e) = 0 \; \forall e \in E$ as the input of the *MLCS* clustering. As quality function for a cluster $C = (O, S)$ we choose $Q(C) = \begin{cases} |O| & |O| \geq k \wedge \gamma_{G_1}(O) = 1 \\ -1 & else \end{cases}$. Since the MLCS clustering has maximum quality sum it contains only the cliques ($\gamma_{G_1}(O) = 1$) of the graph G with at least k vertices. The answer to the k-clique problem is 'no', if the MLCS clustering is empty, and 'yes', else. This proof can even be extended to show the #P-hardness of MLCS. □

3.3 Instantiation of our model

In this section we introduce a default cluster quality function for MLCS clusters. In most settings, clusters containing many vertices are considered more interesting than smaller ones. Therefore, approaches for mining quasi-cliques mostly aim at finding maximal quasi-cliques w.r.t. the number of vertices. However, just maximizing the number of vertices in a cluster can lead to the detection of low-dimensional clusters with low density. Thus, our quality function realizes a trade-off between the contradicting objective functions size, dimensionality and density. Furthermore, we are not interested in clusters that are too small (here: less than 8 vertices) or that are only one-dimensional. Thus, the quality of a cluster $C = (O, S)$ in our instantiation is defined as

$$Q(C) = \begin{cases} |O| \cdot |S| \cdot \gamma_S(O) & |O| \geq 8 \wedge |S| \geq 2 \\ -1 & else \end{cases}$$

Clusters that are not considered interesting are assigned a quality of -1 and will thus never be included in an MLCS clustering, as they would lower the overall quality sum of the clustering. As distance function for the edge labels we use the Manhattan distance. In all experiments in Section 5, these instantiations are used. Though, our model and the algorithm can easily be used with other instantiations, which might be more applicable for some applications.

4. ALGORITHM

In this section we give an overview of the MiMAG (**Mi**ning **M**ulti-layered, **A**ttributed **G**raphs) algorithm. Due to Theorem 1, we cannot expect to find an efficient algorithm computing an exact MLCS clustering. Thus, MiMAG computes an approximate solution: Instead of determining a redundancy-free clustering with *maximum* quality, we compute a *maximal*, redundancy-free clustering with *high quality*. That is, we determine a clustering to which no further cluster C with $Q(C) > 0$ can be added without violating the redundancy-freeness property.

MiMAG is partly based on the Quick algorithm [12] for finding quasi-cliques. In this algorithm, vertex sets $O \subseteq V$ are enumerated by a depth-first traversal in the *set enumeration tree* [18].[1] Each set visited by the depth-first traversal is tested for the quasi-clique property. An exemplary tree for a graph with three vertices is shown in Fig. 3 (top left). Each node O is associated with a candidate set $cand_O$, which contains all vertices that are ordered behind the vertices in O in a given order $\prec$. A child node O'' extends its parent node O by adding one of the vertices from $cand_O$. Basically, the set enumeration tree contains all possible vertex sets $O \subseteq V$. However, the search space can be reduced: If O's candidate set contains a vertex v that can never be part of a quasi-clique $O' \supset O$, we can delete v from the candidate set. For example, in Fig. 3 (top left) if the vertex v_2 is deleted from the candidate set of O, the subtree rooted at

[1] To avoid confusion, we use the term "vertex" for a vertex in the original graph and the term "node" for the nodes of the set enumeration tree, which represent *sets* of vertices.

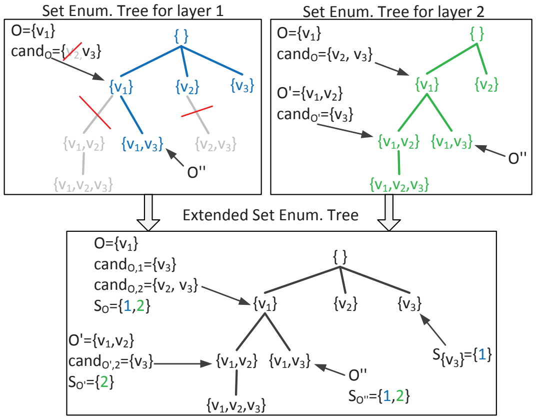


Figure 3: Synchronizing set enumeration trees

$\{v_1, v_2\}$ is pruned from the tree. Techniques how to detect such vertices were introduced in [12].

Synchronized Tree Traversal. A naive approach to determine the MLCS clustering would be: (1) use the quick algorithm on each of the graph layers individually to find all one-dimensional MLCS clusters[2], (2) compose the resulting patterns to multidimensional clusters, and (3) remove redundant clusters. This naive, *sequential* approach, however, is not suitable for the detection of the MLCS clustering: too many (intermediate) patterns are generated which anyway would not be included in the final result due to their redundancy. Thus, we interweave all steps.

We first combine the steps (1)+(2) by proposing a "synchronized traversal" of *all* set enumeration trees *simultaneously*, i.e. all instances of the tree perform the *same* order of traversal. Trees in which a node O was pruned temporarily pause their traversal. Another view on this synchronized traversal is that we use an *extended set enumeration tree* (cf. Fig. 3, bottom). In this tree, each node O has a set of *active dimensions* S_O (which represent the set of dimensions in which the node O has not been pruned from the set enum. tree) and candidate sets $cand_{O,i}$ for each dimension $i \in S_O$. For each set O we visit during the traversal, we check if O forms an MLCS cluster in a subset of its active dimensions. Please do not confuse the active dimensions S_O of a node O and the subspace S of the potential cluster $C = (O, S)$; it holds $S \subseteq S_O$ but the sets are not necessarily equal. We show in Sec. 4.1/4.2 how the set of active dimensions can be used to prune the tree.

Informed Best-First Traversal. We now combine the steps (1)-(3): Instead of first generating all clusters, we let the final (redundancy-free) clustering grow incrementally. Since we want to maximize the quality of the overall clustering, we aim at generating the clusters in decreasing order of their quality and adding the non-redundant clusters with highest quality to the result first. In this case, it is crucial to use a good traversal strategy for the extended set enum. tree.[3] Therefore, we propose an informed best-first traversal: For each node O, we compute a quality estimation that provides an *upper bound* for the maximal quality of any cluster that can be found in the subtree rooted at O. We start the traversal at the root node and in each search step we expand the node O having the highest estimated quality (i.e. MiMAG descends one step into the subtree rooted at O).

Algorithm 1 MiMAG: Best-first search for MLCS clusters

```
Require: ML-Graph $\mathcal{G} = \{G_i \mid i \in Dim\}$ with $G_i = (V, E_i, l_i)$
Ensure: Redundancy-free, maximal clustering $Result$
 1: $Result := \emptyset$
 2: $queue := \{(\emptyset, Dim, \{cand_{\emptyset,i} = V \mid i \in Dim\})\ \}$
 3: while $queue \neq \emptyset$ do
 4:     $Obj := queue.\text{pop}()$
 5:     if $Obj$ is cluster $C = (O, S)$ then
 6:         if $\neg\exists C' \in Result : C \prec_{red} C'$ then $Result.add(C)$
 7:     else                ▷ $Obj$ is $ST = (O, S_O, \{cand_{O,i} \mid i \in S_O\})$
 8:         $neighbors := \bigcup_{i \in S_O} \{v \in cand_{O,i} \mid \exists x \in O : (x, v) \in E_i\}$
 9:         $u := \arg\max_{v \in neighbors} \{\sum_{i \in S_O} deg^{O}_{G_i}(v)\}$
10:         EXPAND$(O, u, S_O, \{cand_{O,i} \mid i \in S_O\})$
11: return Result
12: procedure EXPAND$(O, u, S_O, \{cand_{O,i} \mid i \in S_O\})$
13:     $O_{next} := O \cup \{u\}$, $S_{O_{next}} := \{i \in S_O \mid u \in cand_{O,i}\}$
14:     for all $i \in S_{O_{next}}$ do $cand_{O_{next},i} := cand_{O,i} \setminus \{u\}$
15:     Prune $S_{O_{next}}$ and $cand_{O_{next},i}$ $(\forall i \in S_{O_{next}})$
16:     $ST_{next} = (O_{next}, S_{O_{next}}, \{cand_{O_{next},i} \mid i \in S_{O_{next}}\})$
17:     if $Q_{est}(ST_{next}) \geq 0$ then $queue.\text{insert}(ST_{next})$
18:     for all $i \in S_O$ do $cand_{O,i} := cand_{O,i} \setminus \{u\}$
19:     Prune $S_O$ and $cand_{O,i}$ $(\forall i \in S_O)$
20:     $ST_{remain} = (O, S_O, \{cand_{O,i} \mid i \in S_O\})$
21:     if $Q_{est}(ST_{remain}) \geq 0$ then $queue.\text{insert}(ST_{remain})$
22:     if $\exists$ cluster $C = (O_{next}, S)$, $S \subseteq S_{O_{next}}$ then
23:         if $\neg\exists C' \in Result : C \prec_{red} C'$ then $queue.\text{insert}(C)$
```

One important aspect has to be considered: Even if a cluster C is found at the currently expanded node, it can not be added to the result directly. Since the quality estimation *upper bounds* the quality of the *subtree*, C itself might have a lower quality. Thus, there might exist other subtrees (and potential clusters) with higher (estimated) qualities. Therefore, MiMAG maintains a priority queue which contains the set of subtrees that are still to process (similar to the list $OPEN$ in best-first search) as well as the set of already detected clusters that could not be added to the result so far. This queue is sorted by the (estimated) quality values of the subtrees and clusters. If the first element of the queue is a cluster, no better clusters exist; in this case (and if the cluster is non-redundant to previously selected clusters), we can finally add it to the result set.

In the queue, a subtree (short: ST) is represented by a 3-tuple $ST = (O, S_O, \{cand_{O,i} \mid i \in S_O\})$ where O is the vertex set in the root node of ST, S_O is the set of active dimensions for O and $cand_{O,i}$ are the candidate sets. $Q_{est}(ST)$ denotes the upper bound for the quality of clusters of this subtree. We discuss these upper bounds in Sec. 4.1.

Overall Processing Scheme. The processing of MiMAG is shown in Algorithm 1. Given the input multi-layer graph $\mathcal{G}$, MiMAG computes a redundancy-free, maximal clustering $Result$. Initially, the set $Result$ is empty (line 1); it will be iteratively filled during the processing. At the beginning, the queue contains one element which represents the root node of the extended set enum. tree (line 2). As long as the queue contains elements, the object with the highest (estimated) quality is taken from the queue. If the object is a cluster, no cluster with higher quality can be found anymore, thus we add it to the result set if it is not redundant w.r.t. an

[2] Please note that for each layer we get a different set enum. tree (cf. Fig. 3, top) as different subtrees might be pruned.

[3] If one is interested in generating *all* patterns, an arbitrary traversal strategy can be used. We, however, want to determine only a *subset* of the patterns (the non-redundant, high quality ones).

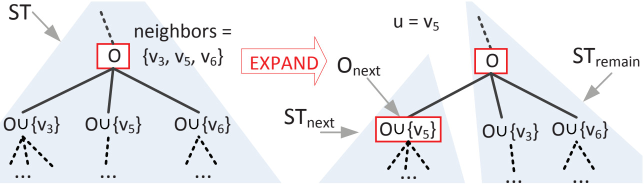


Figure 4: Expansion of a node O

already selected cluster (line 6). If the object is a subtree, we expand the represented set O by one neighboring vertex u that is contained in the candidate sets; we use the vertex having the highest degree w.r.t. O since it most probably leads to dense subgraphs.

The node expansion is illustrated in Fig. 4, where $u = v_5$. MiMAG calls the $EXPAND$ procedure for the subtree rooted at O. In this procedure, at first the sets O_{next}, $S_{O_{next}}$ and the candidate sets $cand_{O_{next},i}$ are determined. $S_{O_{next}}$ can only contain dimensions i for which vertex u was contained in the candidate set $cand_{O,i}$ (line 13). The candidate sets are reduced using pruning techniques (cf. Section 4.2). These sets represent the new subtree ST_{next} rooted at O_{next}, and it is added to the queue if the estimated quality is non-negative (lines 16,17). Similar steps are done for the remaining subtree ST_{remain} rooted at O (lines 20,21), which contains the sets $O' \supset O$ with $u \notin O'$ (cf. Fig. 4). Note that we get a new quality estimate, since u is removed from the candidate sets $cand_{O,i}$. Finally, if O_{next} is a valid (non-redundant) cluster it is also added to the queue.

4.1 Quality Bounds for Subtrees

In the following, we present upper bounds for the quality of subtrees (all proofs are available on our website[4]). Even by allowing arbitrary quality functions, we can derive some generally applicable bounds. First, we exploit the fact that in some cases the subtree does not contain any interesting cluster at all; the quality can be upper bounded by -1.

The first case uses the active dimensions: If no active dimensions are left in the node O, we know that there cannot exist any valid cluster in the subtree rooted at O. This result holds since a cluster's subspace is a subset of the active dimensions and the active dimensions fulfill the anti-monotonicity property[5]: If a dimension i is not active for the set O, then there cannot exist a superset $O' \supset O$ such that i is active for O'.

In the second case, we exploit our redundancy model to determine the bound: If all clusters C contained in subtree ST (i.e. clusters $C = (X, S_X)$ with $S_X \subseteq S_O$ and $O \subset X \subseteq O \cup \bigcup_{i \in S_O} cand_{O,i}$) would be redundant w.r.t. a cluster $C' \in Result$ we cannot add them to the final clustering. Thus, even if their quality is larger than 0, we can safely estimate the subtree's quality with -1. To check the redundancy w.r.t. a cluster $C' = (O', S') \in Result$, we have to check the properties from Def. 5. The properties $C \neq C'$ and $Q(C) \leq Q(C')$ are trivially fulfilled for every possible C due to the ordering of the queue; just the edge overlap property ($\frac{1}{|S_X|} \sum_{i \in S_X \cap S'} \frac{|E_i(X) \cap E_i(O')|}{|E_i(X)|} \geq r$) has to be checked. Therefore, we determine a lower bound ovl_{min} for the edge overlap such that $ovl_{min} \leq \frac{1}{|S_X|} \sum_{i \in S_X \cap S'} \frac{|E_i(X) \cap E_i(O')|}{|E_i(X)|}$ for all possible clusters C from the subtree . Then, if $ovl_{min} \geq r$ we get $Q_{est}(ST) = -1$. For every subtree ST and every cluster $C' = (O', S') \in Result$ with $S' \supseteq S_O$ we get:

$$ovl_{min} = \min_{i \in S_O} \frac{|E_i(O \cap O')| + \max\{\frac{1}{4} \cdot (|O|^2 + |O|) - |E_i(O \cap O')| - k, 0\}}{|E_i(O \cap O')| + k + \max\{\frac{1}{4} \cdot (|O|^2 + |O|) - |E_i(O \cap O')| - k, 0\}}$$
$$\text{with } k = |E_i(O \cup cand_{O,i}) \backslash E_i((O \cup cand_{O,i}) \cap O')|$$

For example, in the case $O' \supseteq O \cup \bigcup_{i \in S_O} cand_{O,i}$ we get $ovl_{min} = 1$ and thus $Q_{est}(ST) = -1$.

Upper bounding cluster properties. Useful properties to incorporate in quality functions are the density and cardinality of clusters. Thus, we develop upper bounds for these cluster properties that can be used for specific instantiations of the quality function. Given a subtree $ST = (O, S_O, \{cand_{O,i} \mid i \in S_O\})$, for each one-dimensional MLCS cluster X in dimension $i \in S_O$ with $O \subset X \subseteq O \cup cand_{O,i}$ the following bounds apply:

- $\gamma(X) \leq \min\{\frac{min\_deg_i}{|O|}, 1\} =: \gamma_i^{max}$ with
  $min\_deg_{G_i} = \min_{v \in O}\{deg^{O \cup cand_{O,i}}(v)\}$
- $|X| \leq \min(\left\lfloor \frac{min\_deg_i}{0.5} \right\rfloor + 1, |O \cup cand_{O,i}|) =: n_i^{max}$
- $|E_i(X)| \leq |E_i(O)| + (n_i^{max} - |O|) \cdot \max_{v \in cand_O} \{deg_{G_i}^{O \cup cand_{O,i}}(v)\}$

Furthermore, we have for each multi-dimensional MLCS cluster (X, S_X): $|S_X| \leq |S_O|$ due to the anti-monotonicity of the active dimensions.

Specific instantiation. We can use the above bounds for our default instantiation of the quality function: the quality of the subtree ST is upper bounded by

$$Q_{est}(ST) = \max_{k \in \{1, \ldots, |S_O|\}} \{max_k(n_i^{max}) \cdot \sum_{m=1}^{k} max_m(\gamma_i^{max})\}$$

where $max_x(y_i)$ denotes the x-th highest value of all $\{y_i \mid i \in S_O\}$. Furthermore, if we have $\max_{i \in S_O}(n_i^{max}) < 8$ or $|S_O| < 2$, the subtree can not contain any cluster with positive quality; in this case, the estimation is $Q_{est}(ST) = -1$.

4.2 Pruning Techniques

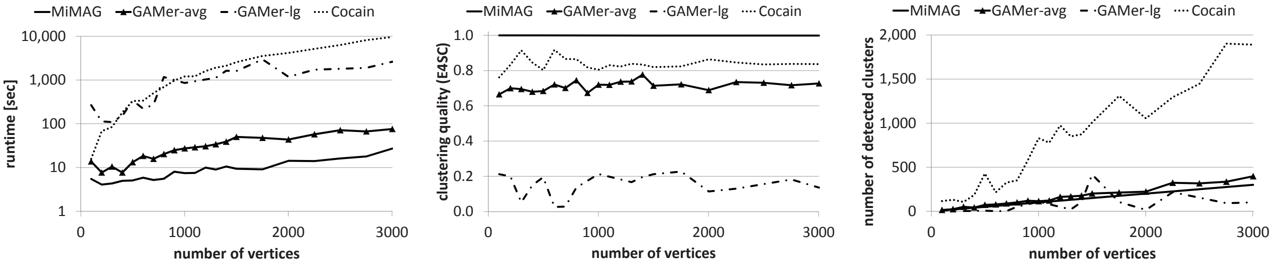
MiMAG exploits the introduced quality bounds twofold: first, to realize the best-first traversal using a priority queue; and second, to prune the search space if the estimate is negative (lines 17, 21). To further enhance the efficiency, MiMAG exploits pruning techniques for the set of active dimensions and the candidate sets (lines 15, 19).

One example is the *pruning by edge similarity*: Due to Def. 3, a one-dim. MLCS cluster (O, S) must only contain edges with similar labels. Thus, if the set $E_i(O)$ contains *any* two edges with label distance greater than w, O (and also all supersets $O' \supset O$) cannot be a valid cluster. We use this property to prune the candidate sets $cand_{O,i}$ as follows: If for a vertex $v \in cand_{O,i}$ it holds that $E' = E_i(O) \cup \{(v, o) \in E_i \mid o \in O\}$ does not fulfill the similarity property, we can remove v from $cand_{O,i}$ as no set $O' \supseteq O \cup \{v\}$ could form an MLCS cluster in dimension i.

Deleting a vertex from a candidate set can change properties (e.g. the degree) of other vertices from the set, thus we prune the sets iteratively until no more vertices can be deleted. If after the pruning we have $cand_{O,i} = \emptyset$ for a dimension i, i becomes inactive and can thus be removed from

[4] http://dme.rwth-aachen.de/mlcs

[5] Note: This property does not hold for the cluster model itself (neither for the set of vertices nor for the relevant dimensions).



(a) Runtime vs. graph size (b) Clustering quality vs. graph size (c) #Detected clusters vs. graph size

Figure 5: Experimental evaluation on synthetic datasets (1)

S_O. More pruning techniques are left out here due to space limitations. Besides the pruning techniques developed especially for the MLCS model, MiMAG also uses the pruning techniques from the Quick algorithm [12].

5. EXPERIMENTAL EVALUATION

We evaluate the clustering quality and runtime of MiMAG experimentally on synthetic and real-world datasets. All experiments were conducted on Opteron 2.3 GHz CPU's using Java6 64bit. For the synthetic data, the clustering quality is determined by comparing the clustering results to the ground truth using the E4SC measure, which was developed for the evaluation of subspace clustering results [5]. For the real-world datasets, there is no ground truth available, which hinders an evaluation of the clustering quality. Thus, for those datasets we provide some key characteristics of the clustering results as well as exemplary clusters to illustrate the results of MiMAG. If not specified otherwise, the redundancy parameter r is set to $r = 0.25$ for all experiments.

Baseline approaches. We compare MiMAG with 3 baseline approaches: The closed quasi-clique mining algorithm Cocain [21] (cf. Section 2) is used on our input graph by considering each of the graph layers as a graph in a graph database. To best match our cluster model, the minimum support parameter of Cocain is set to $min\_sup = \frac{1}{|Dim|}$ and the minimum quasi-clique density to $\gamma_{min} = 0.5$.

Furthermore, we present two different ideas to adapt the GAMer algorithm [7], developed for clustering graphs with vertex labels, to our problem. Both ideas transform the multi-layer graph covering Dim layers to a graph with Dim-dimensional attribute vectors at the vertices. The resulting graph can then be clustered by GAMer. In the first idea (GAMer-avg), the transformed graph is obtained as follows: the vertices of the original graph are kept; the edges are determined by the union of the edge sets from all graph layers (i.e. $E = \bigcup E_i$; the edge labels are deleted). The i-th entry of a vertex v's attribute vector is the average label value of v's incident edges from layer i. In the second idea (GAMer-lg) we use the well-known concept of the *line graph* [8]. Each vertex of a line graph represents an edge of the original graph and vice versa. In our case, a line graph vertex $v^{lg} = (v_1, v_2)$ represents *all* the edges (v_1, v_2) from the different layers. The i-th entry of v^{lg}'s attribute vector corresponds to the label value $l_i(v_1, v_2)$, if $(v_1, v_2) \in E_i$ and $\perp$, else (where $\perp$ is considered not similar to any value).

5.1 Evaluation on synthetic graphs

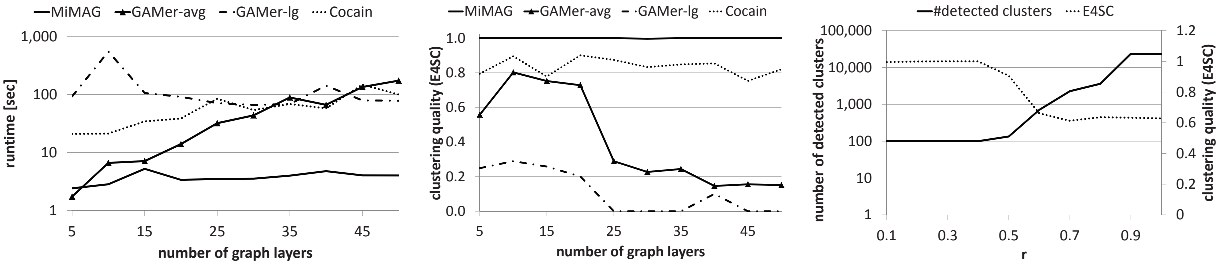
For the evaluation of MiMAG, we generated various synthetic multi-layer graphs with edge labels containing overlapping MLCS clusters as well as "noise" vertices and "noise" edges that do not belong to any cluster. The generated edge labels lie in the range $[0, 1]$. In our experiments, the parameter w for MiMAG (and also the corresponding parameter for GAMer) is set to $w = 0.1$.

Results for varying graph sizes. First, we analyze the behavior of the approaches for varying graph sizes. The generated graphs consist of 10 layers, the number of generated ("hidden") clusters increases linearly from 10 to 300. Each cluster contains 10 vertices and 3 relevant layers, with quasi-clique densities of 0.6. 10% of the vertices in the graph are noise vertices and in each layer we have 60 noise edges.

Although the runtimes of all approaches (cf. Fig. 5(a)) increase with increasing graph sizes, MiMAG constantly shows the lowest runtimes. Considering the clustering quality (cf. Fig. 5(b)), MiMAG reaches perfect or nearly perfect E4SC values on all datasets. The number of detected clusters (Fig. 5(c)) matches the number of hidden clusters. Cocain also achieves quite good quality values (ca. 0.8 to 0.9), as the closed quasi-clique model is closely related to our MLCS model. However, Cocain outputs a huge amount of quasi-cliques (e.g. nearly 2000 instead of the hidden 300 clusters) because it does not avoid redundancy in the result. This also explains the high runtimes of Cocain. For GAMer-avg, the number of detected clusters approximately matches the number of hidden clusters. However, the clustering quality is significantly lower as MiMAG's as the averaged label values distort the cluster structure. For GAMer-lg the number of found clusters varies very much and the clustering quality is low. This is caused by an important problem with the line graph approach: from the density of a subgraph in the line graph, it is not possible to draw conclusions about the density of the corresponding original subgraph, which hinders the detection of dense subgraphs in the original graph.

Results for varying dimensionality. Next, we analyze the behavior of the different approaches for varying dimensionalities (i.e. varying numbers of graph layers) of the input graph. The number of graph layers varies between 5 and 50. The generated multi-layer graphs each contain 30 clusters, each having 10 vertices and 3 relevant layers, with quasi-clique densities of 0.6. Again, we have 10% noise vertices and 60 noise edges per layer.

Comparing the runtimes and clustering qualities of the approaches (Fig. 6(a) and Fig. 6(b)), we observe that MiMAG again achieves the lowest runtime and highest quality. For most approaches, the runtime and the clustering quality remain relatively stable for increasing dimensionality. Just for GAMer-avg the runtime significantly increases, while the E4SC values dramatically drop. This is caused by the graph



(a) Runtime vs. dimensionality (b) Clustering quality vs. dim. (c) #Detected clusters vs. parameter r

Figure 6: Experimental evaluation on synthetic datasets (2)

transformation: As the edges of the transformed graph are the union of the edge sets from all graph layers, by combining an increasing number of graph layers the transformed graph gets very dense, such for high dimensionalities GAMer-avg detects many clusters that do not exist in the original graph.

Results for varying redundancy parameter. In Fig. 6(c), we analyze how the redundancy parameter r of our clustering model affects the results of MiMAG, using a graph with 10 layers and 100 hidden clusters; the cluster size varies between 10 and 15. We observe that for $r < 0.5$, the correct number of clusters is found with a high clustering quality. For $r \geq 0.5$, the number of found clusters increases dramatically and the clustering quality drops. For high values for r, less clusters are considered redundant w.r.t. other clusters, which leads to a clustering that contains many low-quality clusters that would be considered redundant for lower r values. Thus, we propose using $r = 0.25$ as a reasonable default setting for r.

5.2 Evaluation on real-world data

Besides synthetic graphs, we also evaluate our approach on three real-world datasets: The first one is a multi-layer graph with edge labels extracted from the IMDB movie Database[6]. In this graph, the vertices represent actors; the labeled edges represent information about movies in which the actors worked together. The four layers of the graph are: 1. "First year of collaboration", 2. "Last year of collaboration", 3. "Rental fees" (the average earnings of all joint movies between two actors), 4. "Sold tickets" (the average number of sold tickets of all joint movies between two actors). All label values were normalized to the range $[0, 1]$, and we used $w = 0.03$ for this experiment. In this special case, the same edge sets exist in all layers; though, the edge labels differ. Overall, the IMDB graph contains 300 vertices (the most prolific actors) and 18368 edges. An exemplary cluster from MiMAG's clustering result is shown in Fig. 7. Please note that the cluster does not form a clique (its quasi-clique density is 0.625), thus not all actors worked together in the same movie. Actually, all actors connected by an edge worked together (among other movies) in the movie "Con Air" or "The Rock" (or both).

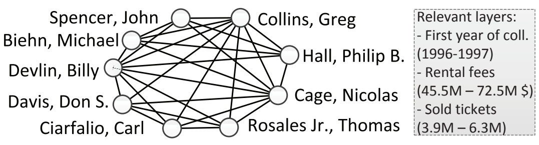


Figure 7: Exemplary cluster from IMDB

In our next experiment, we evaluate the potential of our approach to handle also multi-layer graphs without edge labels. The second dataset was constructed from an extract of the Arxiv publication database[7]. Here, each vertex represents a publication. From the abstracts of the publications, we extracted the 300 most common keywords. Each layer of the graph represents a certain keyword, and an edge of layer i represents a citation between two publications with the common topic i. Overall, the Arxiv graph contains 13396 vertices and 673800 edges. For example, the largest cluster found by MiMAG consists of 19 papers from the field of string theory. The 7 relevant layers of this cluster correspond to the keywords given in Fig 8:

symmetry	model	supersymmetric	intersect
brane	super	metric	

Figure 8: Keywords for a cluster from Arxiv

Our third real-world dataset is a co-author graph extracted from the DBLP database[8]. In this graph, the vertices represent authors and the layers represent the 50 conferences in computer science having the most publications. Two authors are connected by an edge in layer i if they co-authored at least two papers that were published at the corresponding conference. Overall, the DBLP graph contains 17291 vertices and 22896 edges. As we expect co-author groups to be rather small, for this experiment we adapted our quality function to consider clusters with at least 4 vertices as interesting. Fig. 7 shows three exemplary clusters detected by MiMAG and their corresponding conferences. Please note that each of the clusters has different relevant layers. While two of the clusters form cliques in both of their layers, in the top right cluster the edge sets of the layers differ.

Clustering results on real-world datasets. In Table 1, we summarize key characteristics of the clustering results of the different approaches on the real-world datasets. Experiments that did not finish within 2 days were aborted. For each approach and dataset we provide the runtime as well as the average number of vertices, density and number of layers of the found clusters. Note that for the adaptions of GAMer, the density and subspace are determined on the corresponding *transformed* graphs (whose densities are generally higher than in the original graph); the clusters do *not* correspond to MLCS clusters in the *original* multi-layer graphs.

[6] http://imdb.com

[7] http://www.cs.cornell.edu/projects/kddcup/datasets.html

[8] http://dblp.uni-trier.de

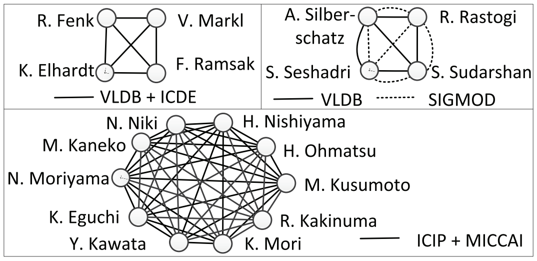


Figure 9: Exemplary clusters from DBLP

Cocain did not finish on any of the datasets within 2 days; GAMer-lg finished only on the DBLP graph, however with a much higher runtime than the other approaches due to the size of the constructed line graph. MiMAG and GAMer-avg have similar runtimes for all datasets. On the IMDB graph, MiMAG detects clusters with a significantly higher average density and dimensionality than GAMer-avg. On Arxiv, the density of GAMer-avg's clusters is slightly higher, which is caused by the fact that GAMer-avg unions the edge sets from all 300 layers and thus obtains a very dense graph. On DBLP, the clusters detected by MiMAG again show the highest average density, while having similar average size and dimensionality to the other approaches.

	IMDB		Arxiv		DBLP		
	MiMAG	*G.-avg*	*MiMAG*	*G.-avg*	*MiMAG*	*G.-avg*	*G.-lg*
runtime [sec]	22	26	623	661	6	9	2390
$avg(\|O\|)$	9.42	9.17	13.6	15.0	4.28	4.48	6.29
$avg(\gamma_S)$	0.94	0.64*	0.62	0.65*	0.87	0.40*	0.81*
$avg(\|S\|)$	2.58	2.00	9.00	9.00	2.05	2.17	2.01

Table 1: Key characteristics of the clustering results (* density in the transformed graph)

6. CONCLUSION

We proposed the new paradigm of clustering multi-layer graphs with edge labels. Besides the mere graph data, additional information about the edges is considered for finding coherent subgraphs. We introduced the clustering model MLCS, which defines clusters of vertices that are *densely connected* by edges with *similar edge labels* in a *subset* of the graph layers. Redundancy in the result set is avoided by selecting only the most interesting clusters. Based on this model, we introduced the efficient best-first search algorithm MiMAG. The performance and clustering quality of MiMAG were demonstrated in our experimental analysis.

Acknowledgments. This work has been supported by the B-IT Research School of the Bonn-Aachen International Center for Information Technology and the UMIC Research Centre, RWTH Aachen University, Germany.

7. REFERENCES

[1] C. Aggarwal and H. Wang. *Managing and Mining Graph Data*. Springer, New York, 2010.

[2] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is ”nearest neighbor” meaningful? In *ICDT*, pages 217–235, 1999.

[3] L. Cerf, T. B. N. Nguyen, and J.-F. Boulicaut. Discovering relevant cross-graph cliques in dynamic networks. In *ISMIS*, pages 513–522, 2009.

[4] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[5] S. Günnemann, I. Färber, E. Müller, I. Assent, and T. Seidl. External evaluation measures for subspace clustering. In *CIKM*, pages 1363–1372, 2011.

[6] S. Günnemann, B. Boden, and T. Seidl. DB-CSC: A density-based approach for subspace clustering in graphs with feature vectors. In *ECML/PKDD (1)*, pages 565–580, 2011.

[7] S. Günnemann, I. Färber, B. Boden, and T. Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *ICDM*, pages 845–850, 2010.

[8] F. Harary and R. Norman. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 9(2):161–168, 1960.

[9] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics*, 4(2):100 –107, 1968.

[10] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 3(1):1–58, 2009.

[11] M. Li, Y. Fan, J. Chen, L. Gao, Z. Di, and J. Wu. Weighted networks of scientific communication: the measurement and topological role of weight. *Physica A: Statistical Mechanics and its Applications*, 350(2):643–656, 2005.

[12] G. Liu and L. Wong. Effective pruning techniques for mining quasi-cliques. In *ECML/PKDD (2)*, pages 33–49, 2008.

[13] E. Müller, I. Assent, S. Günnemann, R. Krieger, and T. Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *ICDM*, pages 377–386, 2009.

[14] F. Moser, R. Colak, A. Rafiey, and M. Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, pages 593–604, 2009.

[15] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. In *VLDB*, pages 1270–1281, 2009.

[16] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison Wesley, 1984.

[17] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *SIGKDD*, pages 228–238, 2005.

[18] R. Rymon. Search through systematic set enumeration. In *KR*, pages 539–550, 1992.

[19] M. Shiga, I. Takigawa, and H. Mamitsuka. A spectral clustering approach to optimally combining numerical vectors with a modular network. In *SIGKDD*, pages 647–656, 2007.

[20] J. Wang, Z. Zeng, and L. Zhou. Clan: An algorithm for mining closed cliques from large dense graph databases. In *ICDE*, page 73, 2006.

[21] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *SIGKDD*, pages 797–802, 2006.