Knowl Inf Syst (2014) 40:243–278
DOI 10.1007/s10115-013-0640-z

REGULAR PAPER

GAMer: a synthesis of subspace clustering and dense subgraph mining

Stephan Günnemann · Ines Färber · Brigitte Boden · Thomas Seidl

Received: 19 March 2012 / Revised: 26 March 2013 / Accepted: 5 April 2013 /
Published online: 8 May 2013

Abstract In this work, we propose a new method to find homogeneous object groups in a single vertex-labeled graph. The basic premise is that many prevalent datasets consist of multiple types of information: graph data to represent the relations between objects and attribute data to characterize the single objects. Analyzing both information types simultaneously can increase the expressiveness of the resulting patterns. Our patterns of interest are sets of objects that are densely connected within the associated graph and as well show high similarity regarding their attributes. As for attribute data it is known that full-space clustering often is futile, we have to analyze the similarity of objects regarding subsets of their attributes. In order to take full advantage of all present information, we combine the paradigms of *dense subgraph mining* and *subspace clustering*. For our approach, we face several challenges to achieve a sound combination of the two paradigms. We maximize our *twofold clusters* according to their density, size, and number of relevant dimensions. The optimization of these three objectives usually is conflicting; thus, we realize a trade-off between these characteristics to obtain meaningful patterns. We develop a redundancy model to confine the clustering to a manageable size by selecting only the most interesting clusters for the result set. We prove the complexity of our clustering model and we particularly focus on the exploration of various

Stephan Günnemann is supported by a fellowship within the postdoc-program of the German Academic Exchange Service (DAAD).

S. Günnemann (✉)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: sguennem@cs.cmu.edu

I. Färber · B. Boden · T. Seidl
Data Management and Data Exploration Group, RWTH Aachen University, Aachen, Germany

I. Färber
e-mail: faerber@cs.rwth-aachen.de

B. Boden
e-mail: boden@cs.rwth-aachen.de

T. Seidl
e-mail: seidl@cs.rwth-aachen.de

pruning strategies to design the efficient algorithm GAMer (**G**raph & **A**ttribute **M**in**er**). In thorough experiments on synthetic and real world data we show that GAMer achieves low runtimes and high clustering qualities. We provide all datasets, measures, executables, and parameter settings on our website http://dme.rwth-aachen.de/gamer.

Keywords Subspace clustering · Dense subgraph mining · Pruning techniques

1 Introduction

The analysis of graph data is an important and challenging data mining topic. Out of the wide range of different graph mining tasks, this work focuses on the problem of identifying groups of vertices within a single large graph, such that the members of a group are strongly related to each other. For a traditional graph description, consisting of a set of vertices and a set of edges between pairs of vertices, this problem is known as *dense subgraph mining* [2]. Here a group of vertices is regarded as highly related if the vertices are densely connected to each other. For example, functionally related genes are grouped together in gene interaction networks, people showing the same friendship relations in social networks, or sensors communicating to each other in sensor networks. The advances in data recording together with a general attempt of collecting as much information as possible have led to heterogeneous databases, which, in our case, go beyond the mere relationship information of objects. In many applications typically a lot of attribute information is available for the objects. Restricting the data representation to, e.g., a vector representation of the attribute information for each object, transforms our original task of finding groups of vertices to the task of *traditional clustering* [15]. Here a group of objects is regarded as highly related if the objects show high similarity to each other regarding their attributes. For example, genes are clustered based on similar expression levels, people in social networks based on their common interests, or sensors in sensor networks according to similar measurements as temperature and humidity. Both information types together can be modeled as a vertex-labeled graph, in which vertices represent objects, edges represent relations between them, and feature vectors associated with the vertices represent the attributes for each object (cf. Fig. 1).

Since each paradigm—dense subgraph mining and traditional clustering—groups the objects based only on a single type of information, namely graph data or attribute data, the resulting groupings might be highly differing and contradicting. Thus, for applications where both data categories are available, a *simultaneous* use of both information types for the process of clustering promises more meaningful and accurate results. Therefore, combined

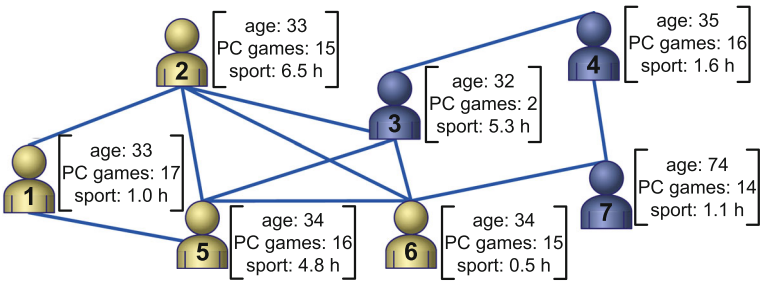


Fig. 1 Combination of graph & attribute data and one potential twofold cluster (highlighted in *yellow*) with two relevant dimensions (color figure online)

clustering approaches have been recently introduced, which try to determine groups that are densely connected within the graph as well as similar according to their attribute values. As shown, e.g., in [16,36], such combined clustering methods can outperform methods using just a single data source.

The main problem of almost all combined clustering approaches, however, is the consideration of *all* attribute dimensions for determining the similarity. Often, some dimensions are not relevant for all clusters, which is why clusters are located in subsets of the dimensions. For example, in social networks, it is very unlikely that people are similar within all of their characteristics. A continuative aspect is the decreasing discrimination power of distance functions with increasing dimensionality of the data space due to the curse of dimensionality [5]. The distances between objects grow more and more alike, thus all objects seem equally similar based on their attribute values. Since clusters are strongly obfuscated by irrelevant dimensions and distances are not discriminable any more, searches in the full-space are futile or lead to very questionable clustering results. As a solution, subspace clustering methods [19,30] were introduced, especially useful for high-dimensional data like genes. Consistently, also combined clustering models should analyze subsets of the attributes. However, this aspect is not adequately considered by the models.

Our novel approach combines graph data and attribute data to identify groups according to their density of connections as well as similarity of their attribute values. In contrast to other approaches, however, we consider subsets of the dimensions to realize meaningful similarity determination. In Fig. 1 for example we are able to identify the cluster $\{1, 2, 5, 6\}$ because the objects are similar in two attributes and the density of the subgraph is high. A clustering procedure like this is advantageous for a variety of applications: Besides the already mentioned example of gene analysis, highly connected groups of people in social networks (graph density) can be used for target and viral marketing based on their specific preferences (attribute subset). In sensor networks, an aggregated transmission of specific sensor measurements (attribute subset) of communicating sensors (graph density) leads to improved energy efficiency and thus longer lifetime of the network.

A sound combination of the paradigms *subspace clustering* and *dense subgraph mining* has to be unbiased in the sense that none of the paradigms is preferred over the other. Most combined clustering models focus on graph properties as determining maximal sets whose density is large enough. In Fig. 1, for example, the largest clique (a certain type of dense subgraphs) is $\{2, 3, 5, 6\}$; however, the vertices of this clique show similar behavior only in one of their three attributes. Even worse, preferring just high-dimensional clusters leads to $\{1, 4, 6\}$; this cluster cannot be reconciled with the graph structure. Obviously, the cluster properties 'density/connectedness', 'dimensionality', and 'size' are usually contradictory. Thus, a clustering model has to realize a reasonable trade-off. The challenge tackled by our approach is the optimization of all three goals simultaneously to ensure their equality. This enables both paradigms to act on an equal footing in order to obtain meaningful and consistent clusters. Vertex group $\{1, 2, 5, 6\}$ and vertex group $\{2, 3, 5\}$ could be possible clusters for such an optimization. In both clusters all vertices have similar values in two attributes and the density of the subgraphs is negligibly smaller than in cliques.

A further important observation is that overlaps between clusters are quite reasonable. While the cluster $\{1, 2, 5, 6\}$ might be of interest for video game producers, the cluster $\{2, 3, 5\}$ might be of interest for sports wear retailers. Persons thus can be assigned to more than one product target group. Also for the application of gene interaction networks and sensor networks it holds that genes can belong to more than one functional module and sensors to more than one aggregation unit. Highly overlapping clusters, however, often imply nearly the same interpretations. Thus, a strong overlap usually indicates redundancy. Especially in

the area of subspace clustering, where for each cluster exponentially many projections in its subspaces exist, considering redundancy is indispensable [24]. Also in the field of graph mining, avoiding redundant patterns is studied [4]. The importance of a proper treatment of redundancy is hence increased for the combined consideration of subspace clustering and subgraph mining. However, this aspect is rarely treated accurately by previous approaches. Our model successfully avoids redundancy in the clustering result, while generally allowing the clusters to overlap.

In this work, we will present our new GAMer (**G**raph & **A**ttribute **M**in**er**) approach, an earlier version of which already appeared in [11]. Besides presenting the theoretical clustering model which we introduced in [11], this article proves important complexity results and focuses on the algorithmic solution of the problem by introducing various pruning techniques. These techniques allow for an efficient computation of our overall clustering model. The main contributions of our work are the following:

- We introduce a novel cluster model, which equitably joins the paradigms of subspace clustering and dense subgraph mining.
- We define a novel clustering model, which includes a redundancy model to avoid unnecessary increase of the result set and at the same time permits overlaps between clusters in general.
- We develop the algorithm GAMer, which exploits various pruning strategies for the efficient calculation of the defined clustering.

Before presenting our approach the following section examines existing approaches and their drawbacks in the analysis of attribute and graph data.

2 Related work

Generally speaking, the aim of clustering is to group objects into so-called *clusters* such that similar objects belong to the same cluster and dissimilar objects belong to different clusters.

Clustering vector data. Traditional clustering for *vector data* evaluates clusters regarding all attributes in the full data space. Independent of the clustering model, they do not scale to high-dimensional data due to the irrelevance of some attributes for individual clusters [5,15]. Since dimensions are mostly not globally irrelevant, which is why global dimensionality reduction techniques like PCA [17] are not applicable, *subspace clustering methods* detect relevant subspace projections for each cluster individually [19,30]. This is also important for our task, because we cannot expect that densely connected groups are similar in all attributes; they belong to different subspaces.

In contrast to traditional clustering algorithms which mostly *partition* the data into clusters, subspace clustering algorithms allow the clusters to *overlap* because in many applications it is possible that one object belongs to several groups, e.g. in different subspaces. However, by allowing overlapping clusters, the problem of *redundancy* arises: If clusters highly overlap, they are very similar to each other. A subspace clustering approach that ignores this fact will output a huge amount of clusters, which nearly represent the same information. For this reason, recent subspace clustering approaches [14,24,27] use *redundancy models* to confine the output to the most interesting clusters and thus get a reasonable result size.

In [28], recent subspace clustering approaches are compared and evaluated. A distinction between cell-based [13,32,35,42], density-based [18], and clustering-oriented [3,25] methods is made. Cell-based approaches have shown to be very efficient and generate high-quality

clusterings. None of the proposed subspace clustering methods, however, considers graph data.

Clustering graph data. Clustering *graph data* has been done in different ways [2]: For our purpose of finding groups in networks, we focus on methods mining densely connected subgraphs in one large graph. Methods as [7,22,23,33] perform a graph partitioning while others assume that the given graph naturally divides into (possibly overlapping) *dense subgraphs*, e.g. cliques [41] or γ-quasi-cliques [1,21,31,45]. None of these approaches, however, considers attribute data for the objects.

As in the area of subspace clustering, by using non-partitioning approaches and allowing the patterns (e.g. quasi-cliques) to overlap, we encounter the problem of redundancy in the output set. For frequent subgraph mining, the problem of redundancy between subgraphs has recently been tackled by some approaches [4,47]. However, in the field of graph clustering/dense subgraph mining the problem of redundancy has not yet been researched thoroughly.

Combined clustering approaches. Apart from the previously mentioned approaches, there exist methods that consider both information types: vector data and graph data. Some of these techniques [8,20] consider attribute data only in a post-processing step after determining subgraphs. Thus, attributes do not influence the resulting subgraph structures. In [16] the network topology is transformed into a (shortest path) distance and is combined with the feature distance such that any distance-based clustering algorithm can be applied afterward. Using combined distance functions like this leads to results that are difficult to interpret as no conclusions about cluster structures in the graph are possible. Similarly, the methods of [29,36] use a weighted combination of the network information with the attribute information but apply the principle of spectral clustering to determine the actual groupings. In contrast to [16], which transforms the graph to distance values, the approach by [40] transforms the feature information into an edge-weighted similarity graph and determines dense subgraphs based on the original graph and the similarity graph. Directly operating on the original data, the method of [9] extends the k-center problem by requiring that each group has to be a connected subgraph. All the previous approaches [9,16,29,36,40] cannot detect similarities between objects based on subsets of their attributes because they use full-space similarity. Since such similarity values are often not discriminable [5], they unconsciously use only the graph information for clustering, making the desired combination meaningless. Furthermore, these approaches determine disjoint, or almost disjoint, clusters. In general, methods performing a graph partitioning, e.g. by exploiting the idea of spectral clustering, are not suited in our scenario since we want to permit overlaps between the clusters. In [48] and [49] categorical attribute values are modeled as additional structural nodes into the original graph. Due to these structural nodes, new paths between vertices with similar feature values arise. Although objects in one cluster do not necessarily show similarity in all attributes, they are only pairwise similar and no specific relevant attribute subset can be defined for the clusters. In [38] graphs with labeled vertices are considered. This approach detects patterns that are quasi-cliques where all vertices have a set of labels in common. However, the approach only considers categorical labels and cannot handle data with continuous attributes. CoPaM [26] is the only approach so far that deals with subspace clustering and dense subgraph mining. Though, it considers the density and the subspace cardinality only as minimal threshold constraints. Since CoPaM solely optimizes the number of vertices, density and subspaces are just incidental. In our approach, however, we balance these measures yielding an unbiased combination of subspace clustering and dense subgraph mining.

Besides the already mentioned disadvantages of all existing combined clustering methods, one major drawback is their missing or limited redundancy handling. Especially, for

approaches analyzing subsets of attributes, as CoPaM for the combined clustering paradigm, this redundancy removal is essential. CoPaM, however, does not consider redundancy of the resulting maximal subgraphs, which thus can potentially overlap to a high extent. In our approach, redundant clusters are removed from the output.

3 A combined clustering model

In this section, we present our model for the detection of densely connected subgraphs that exhibit feature similarity in subsets of the dimensions—called *twofold clusters*. Our model combines subspace clustering with dense subgraph mining. To this end, we model attribute data together with graph data. Formally, the input is a vertex-labeled graph $G = (V, E, l)$ with vertices V, edges $E \subseteq V \times V$ and a labeling function $l : V \rightarrow \mathbb{R}^d$ where $Dim = \{1, \ldots, d\}$ is the set of dimensions. We assume an undirected graph without self-loops, i.e. $(v, u) \in E \Leftrightarrow (u, v) \in E$ and $(u, u) \notin E$. As an abbreviation we use $l(O) = \{l(o) \mid o \in O\}$ to denote the set of vectors associated with the set of vertices $O \subseteq V$ and $x[i]$ to refer to the i-th component of a vector $x \in \mathbb{R}^d$.

We introduce in Sect. 3.1 the cluster definition that defines the properties a single valid cluster has to fulfill. In Sect. 3.2, our clustering criteria are defined, which favor the selection of the most interesting clusters. Since many similar clusters can be valid, this definition is crucial to prevent high redundancy in the output. The complexity of our clustering model is analyzed in detail in Sect. 3.3. We conclude in Sect. 3.4 with a discussion of the model's parameters and we provide guidelines how to set their values.

3.1 Cluster definition

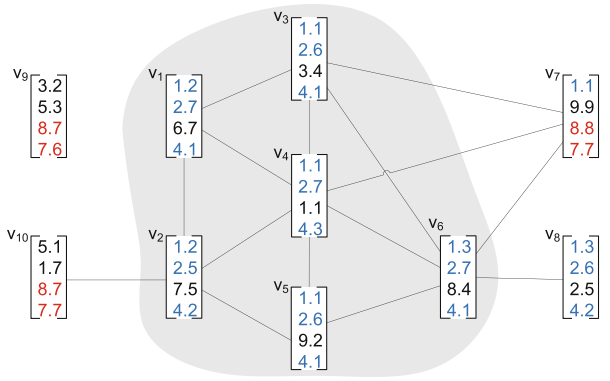
Our twofold clusters should represent meaningful subspace clusters and at the same time meaningful dense subgraphs. Therefore we combine established definitions of both paradigms. For subspace clustering the cell-based methods show high-quality results and they are efficiently computable [28]. Thus, we choose this paradigm for our model and use a cluster definition similar to the one used by [32]. A subspace cluster is a set of objects along with a set of relevant dimensions. Within the relevant dimensions, the objects are very similar, i.e. the variation of their attribute values is restricted to a maximal width w. For the non-relevant attributes the values differ to a higher extent.

Definition 1 (*Subspace cluster property*) Given a set of vectors $X \subseteq \mathbb{R}^d$ and a set of dimensions $S \subseteq Dim$, the tuple (X, S) is a subspace cluster (with respect to the maximal width w) if:

- $\forall i \in S : \forall x_1, x_2 \in X : |x_1[i] - x_2[i]| \leq w$
- $\forall i \in Dim \backslash S : \exists x_1, x_2 \in X : |x_1[i] - x_2[i]| > w$

In Fig. 2, the vectors $l(O_1)$ and the dimensions S_1 are a valid subspace cluster for $w = 0.5$. Another subspace cluster is $(l(O_4), S_4)$. By normalizing attributes, different w values per dimension can be realized and by choosing $w = 0$, categorical data can be analyzed if categories are represented by natural numbers.

For identifying dense subgraphs we use the notion of quasi-cliques [21]. Within a quasi-clique O, each vertex $v \in O$ has to be connected to a certain minimal percentage of vertices of O. This minimal degree reflects the density more accurately than the average degree of the vertices. Furthermore, the strict complete pairwise connectivity as for usual cliques is relaxed with this definition.



objects	relevant dimensions	density
$O_1 = \{v_1, \ldots, v_6, v_8\}$	$S_1 = \{1, 2, 4\}$	$\gamma(O_1) = 0.16$
$O_2 = \{v_1, \ldots, v_6, v_7\}$	$S_2 = \{1\}$	$\gamma(O_2) = 0.5$
$O_3 = \{v_1, \ldots, v_6\}$	$S_3 = \{1, 2, 4\}$	$\gamma(O_3) = 0.6$
$O_4 = \{v_7, v_9, v_{10}\}$	$S_4 = \{3, 4\}$	$\gamma(O_4) = 0$

Fig. 2 Exemplary groups and their properties

Definition 2 (*Quasi-clique property*) A set of vertices $O \subseteq V$ within a graph $G = (V, E, l)$ is a γ-quasi-clique if

$$\min_{v \in O}\{deg^O(v)\} \geq \lceil \gamma \cdot (|O| - 1) \rceil .$$

Here, $deg^O(v)$ is the degree of vertex v within vertex set O, i.e. $deg^O(v) = |\{o \in O \mid (v, o) \in E\}|$. The density of a quasi-clique is defined by:

$$\gamma(O) = \frac{\min_{v \in O}\{deg^O(v)\}}{|O| - 1}$$

In Fig. 2, the set O_2 is a 0.5-quasi-clique with maximal number of vertices. Each vertex is connected to at least 3 other vertices within the group.

Quasi-cliques describe object sets based on their connectivity; these groups are dense but potentially have only few (or even no) relevant dimensions. Contrarily, subspace clusters describe object sets based on their similarity in subspaces; these subspaces are large in general but the underlying subgraph is potentially not dense (or even not connected). For this reason, our twofold clusters have to fulfill both properties simultaneously. We formalize:

Definition 3 *(Twofold cluster)* Given a graph $G = (V, E, l)$, a twofold cluster $C = (O, S)$ with respect to the thresholds $s_{min}, \gamma_{min}, n_{min}$ is a set of vertices $O \subseteq V$ and a set of dimensions $S \subseteq Dim$ with the following properties:

- $(l(O), S)$ fulfills the subspace cluster property with $|S| \geq s_{min}$
- O fulfills the quasi-clique property with $\gamma(O) \geq \gamma_{min}$
- the induced subgraph of O is connected and $|O| \geq n_{min}$

With the three minimum-thresholds we are able to parametrize the requirements for a twofold cluster; these are the properties that are important for subspace clusters and dense subgraphs. Additionally, we have to ensure the connectivity of our vertex sets. If $\gamma_{min} < 0.5$, a quasi-clique does not necessarily need to be connected [45]. For a twofold cluster, however,

the connectivity is reasonable and enforced by our definition. For example, in Fig. 2, we get the twofold cluster (O_3, S_3) by choosing $n_{min} = 3, s_{min} = 2, \gamma_{min} = 0.5$. The previous examples for subspace clusters or quasi-cliques, however, are not valid twofold clusters because at least one of our properties is violated by these sets. With Definition 3, we get more meaningful clusters.

3.2 Clustering definition

With the beforehand introduced definition we are able to determine the set of all valid twofold clusters $Clusters$. Without any constraints this set can be large because we permit overlapping clusters in general (cf. Fig. 1). For example, by choosing $s_{min} = 1$ the cluster (O_2, S_2) in Fig. 2 is also valid. This cluster, however, intuitively provides only little novel information compared to the cluster (O_3, S_3); the vertices differ only marginally and we have less dimensions. By introducing a clustering definition, i.e. by determining a meaningful subset $Result \subseteq Clusters$, we focus on the most interesting clusters. Redundant clusters, which provide only little additional information, are not included in our result. The interestingness of clusters is presented in Sect. 3.2.1, our redundancy model is presented in Sect. 3.2.2. In Sect. 3.2.3, the overall clustering is defined.

3.2.1 Quality function

To confine the $Result \subseteq Clusters$ to the most interesting clusters, a measure for the quality of a cluster is necessary. The interestingness of our twofold clusters cannot be solved trivially. Usually, subspace clustering models try to maximize the dimensionality of clusters while dense subgraph methods maximize either the number of vertices or the density of the subgraph. Optimizing all these properties, however, results in conflicting objective functions. For example it is possible that a set of vertices has a high density and by adding just one vertex to this set, e.g. to achieve a maximal cluster size, the density dramatically drops to a low value. It is thus mandatory to trade off these characteristics of clusters to realize a sound and unbiased synthesis of subspace and subgraph mining. Our quality function rates the interestingness of a twofold cluster based on these three aspects.

Definition 4 (*Quality of a twofold cluster*) Given a twofold cluster $C = (O, S)$, the quality of C is defined by $Q(C) = \gamma(O)^a \cdot |O|^b \cdot |S|^c$.

By this quality function we get a flexible model that is easily adaptable. With $a = c = 0, b = 1$, for example, we account only for the number of vertices as in many other models. With $a = b = c = 1$, however, we rate all aspects equally. For example, in Fig. 2, we thus get for the cluster $C_2 = (O_2, S_2)$ and $C_3 = (O_3, S_3)$ the quality values $Q(C_2) = 0.5 \cdot 7 \cdot 1 = 3.5$ and $Q(C_3) = 10.8$.

3.2.2 Redundancy model

With our redundancy model we identify clusters that contribute only little to the information content of the final result. Previous approaches use the maximality of patterns (with respect to objects or dimensions) to exclude other patterns that correspond to subsets of the objects/dimensions. In our model we have two reasons why this is not meaningful. First, the maximal clusters may differ only in few objects/dimensions as well; in this case, they provide no novel knowledge and the result size can still be large. Second and even more problematic, the maximal clusters are not necessarily the most interesting clusters in our model.

The quality function is important to identify the redundant clusters. A cluster C can only be redundant compared to a cluster C' if the quality of C is lower. If the cluster C had a higher quality, then it should not be reported as redundant with respect to C'; the user is more interested in C. Thus, $Q(C) < Q(C')$ must hold for the redundancy of C with respect to C'.

Furthermore, the cluster C induces redundancy with respect to C' if it does not describe novel structural information. We define a cluster $C = (O, S)$ as structural similar to C' if most of its vertices and relevant dimensions are already covered by the cluster $C' = (O', S')$: If the fraction $\frac{|O \cap O'|}{|O|}$ is large, only a small percentage of C's objects are not contained in C'; we do not have a large information gain based on the object grouping of C. The same holds for the set of relevant dimensions. If all three indicators are valid, the cluster C is redundant with respect to C'. We denote this by $C \prec_{red} C'$ and we formally define:

Definition 5 (*Binary redundancy relation*) Given the redundancy parameters $r_{obj}, r_{dim} \in [0, 1]$, the binary redundancy relation $\prec_{red}$ is defined by: For all twofold clusters $C = (O, S), C' = (O', S')$:

$$C \prec_{red} C' \Leftrightarrow Q(C) < Q(C') \wedge \frac{|O \cap O'|}{|O|} \geq r_{obj} \wedge \frac{|S \cap S'|}{|S|} \geq r_{dim}$$

The parameters r_{obj}, r_{dim} intuitively represent the percentage of already covered objects and dimensions of the cluster C under consideration. Note that our redundancy relation is non-transitive, i.e. we can have clusters $\{C_a, C_b, C_c\}$ with $C_a \prec_{red} C_b, C_b \prec_{red} C_c$ but $\neg(C_a \prec_{red} C_c)$. Furthermore, $\prec_{red}$ is irreflexive, i.e. a cluster is never redundant to itself, and $\prec_{red}$ is asymmetric, i.e. we cannot get $C_b \prec_{red} C_a$ if $C_a \prec_{red} C_b$ holds.

The higher the redundancy parameter r_{obj}/r_{dim}, the more objects/dimensions of C have to be covered by C'. For the extremal case of $r_{obj} = r_{dim} = 1$, C's objects/dimensions have to be a subset of the ones of C'. In this case, only few clusters are redundant. By choosing smaller values, the redundancy occurs more often. Considering the clusters $C_2 = (O_2, S_2)$ and $C_3 = (O_3, S_3)$ in Fig. 2, we get $C_2 \prec_{red} C_3$ for $r_{obj} = r_{dim} = 0.5$. By choosing $r_{obj} = 1, r_{dim} = 0.5$, however, none of the clusters is redundant compared to the other.

To identify a cluster as redundant, all three indicators have to occur. It is not enough that e.g. the set of objects is covered; if considerably many other dimensions are comprised, the cluster will still not be redundant.

3.2.3 Determining overall clustering

Up to now we defined a binary relation for pairwise redundancy of clusters. The final step is to define the overall clustering, i.e. given the set of all twofold clusters $Clusters$ we want to get a meaningful subset $Result \subseteq Clusters$.

Since redundant clusters provide only little novel information, they are not beneficial for the user. Thus, the final clustering has to fulfill the *redundancy-free property*: The result set must not contain clusters of which one is redundant to another. To achieve this property it would be sufficient to remove all clusters from $Clusters$ that are redundant to at least one other cluster. Formally we would get $Result = \{C \in Clusters \mid \neg\exists C' \in Clusters : C \prec_{red} C'\}$. This solution, however, is too naive for our model because our redundancy relation is non-transitive. For the clustering $\{C_a, C_b, C_c\}$, introduced in the non-transitivity discussion above, a removal of all possibly redundant clusters would result in just $\{C_c\}$. However, the result $\{C_a, C_c\}$ is more useful since these clusters are also pairwise non-redundant and we additionally get C_a.

Evidently, we need our result to fulfill a second property – the *maximality property*: For all clusters C not selected for the result set, there is at least one selected cluster to which C is redundant. Thus, if we select C the redundancy-free property would be violated. Our overall clustering result is:

Definition 6 (*Optimal twofold clustering*) Given the set of all twofold clusters $Clusters$, the optimal twofold clustering $Result \subseteq Clusters$ fulfills:

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

The maximality property ensures that our optimal result contains the most interesting clusters. In our previous example with $\{C_a, C_b, C_c\}$, e.g., the clustering $\{C_b\}$ is not optimal. Although, no further cluster can be added without introducing redundancy to the result, the second criterion is violated since $C_a \prec C_b$ does not hold. Thus we get the desired maximal result of $Result = \{C_a, C_c\}$. With our optimal twofold clustering the output is confined to the most interesting clusters and redundant clusters are avoided. Due to the meaningful result size and by incorporating attribute information in subspaces with the paradigm of dense subgraphs, the user is able to extract novel knowledge.

3.3 Complexity analysis

In this section, we analyze the complexity of our clustering model. First we show that the overall complexity of our model, i.e. for generating the twofold clusters *and* selecting the optimal clustering, is #P-hard. We denote this problem as $OVERALL$. Second, however, we show that if the set of twofold clusters $Clusters$ is already given, selecting the optimal clustering $Result \subseteq Clusters$ can be done in polynomial time. We denote this subproblem as $SELECT$.

3.3.1 Complexity of the overall result determination

Theorem 1 *Given a vertex-labeled graph $G = (V, E, l)$, determining the optimal clustering according to Def. 6 is #P-hard with respect to the number of vertices V.*

We prove the #P-hardness of our overall result determination ($OVERALL$) by a polynomial reduction from the #P-hard problem $MAXCLIQUE$ [10] of finding the number of all maximal cliques with at least k nodes in a graph $G = (V, E)$. We prove that $OVERALL$ can be used to solve $MAXCLIQUE$, i.e.

$$MAXCLIQUE \leq_P OVERALL$$

Proof ***Input Mapping:*** We use the original graph G and choose $\gamma_{min} = 1, n_{min} = k, s_{min} = 0, r_{obj} = 1, r_{dim} = 0, a = 0, b = 1, c = 0$.

Output Mapping: The cardinality of the result $Result$ obtained by $OVERALL$ corresponds to the number of maximum cliques in the graph.

(1): The set of twofold clusters only contains all cliques ($\gamma_{min} = 1$) of at least size k ($n_{min} = k$). As for usual cliques, the attribute values do not matter ($s_{min} = 0$).

(2): Only subsets of clusters induce redundancy, i.e.

$$\begin{aligned} C = (O, S) \prec_{red} C' = (O', S') \\ \Leftrightarrow Q(C) < Q(C') \wedge \frac{|O \cap O'|}{|O|} \geq 1 \wedge \frac{|S \cap S'|}{|S|} \geq 0 \\ \Leftrightarrow |O| < |O'| \wedge O \subseteq O' \Leftrightarrow O \subset O' \end{aligned}$$

Consequently, for a maximal cluster C_m there exists no C with $C_m \prec_{red} C$. Accordingly, for each non-maximal cluster C_n, there exists a maximal cluster with $C_n \prec_{red} C_m$.

(3): Since the optimal twofold clustering is maximal and redundancy-free, $Result$ contains all but only the maximal clusters, which correspond to the maximal cliques. Thus, $OVERALL$ generates a valid solution for $MAXCLIQUE$. □

3.3.2 Complexity of selecting the final clustering

While the overall complexity of our model is #P-hard, we can show that the subproblem of redundancy elimination can be solved efficiently.

Theorem 2 *Given a set of twofold clusters $Clusters$, the optimal clustering according to Def. 6 can be determined in quadratic time with respect to the number of clusters.*

To prove this theorem we provide an algorithm calculating the optimal result. We assume that the set of all valid clusters $Clusters$ is given. If a cluster C is not selected for the result, there must exist another cluster C_r in the result with $C \prec_{red} C_r$ (cf. maximality). Particularly, C_r must have a higher quality than C. As a consequence our clustering result always contains the most interesting cluster denoted by C_1^+. Since this cluster has the highest quality, it cannot be redundant to another cluster. Furthermore, all clusters that are redundant to C_1^+ cannot be selected for the result and thus can be removed from our set of clusters $Clusters$. From the remaining clusters again the cluster C_2^+ with the highest quality has to be selected for our result. This cluster is not redundant to C_1^+ (the redundancy-free property is still fulfilled) and also not redundant to all remaining clusters (they have lower quality). To ensure the maximality, we select C_2^+ and we can remove all clusters being redundant to C_2^+. These steps are repeated until no clusters remain.

Thus, our optimal twofold clustering can be calculated with the procedure that is illustrated in Fig. 3. At the beginning, our result set is empty ($Result = \emptyset$) and clusters are ranked in descending order based on their quality values. In each step, we remove the first cluster C from the queue. If there exists a cluster $C' \in Result$ with $C \prec_{red} C'$, the cluster is rejected. Otherwise, we add C to $Result$ and select the next cluster. Based on this procedure we can infer the following corollary.

Corollary 1 *The clustering result based on Definition 6 contains the most interesting (top ranked), non-redundant clusters. This set of clusters is unique.*

To analyze the worst case runtime complexity of the algorithm, we can distinguish two phases: In the first phase we have to sort the clusters, i.e. the complexity is $\mathcal{O}(|Clusters| \cdot log(|Clusters|))$. In the second phase, each cluster is compared to the current result set to evaluate the redundancy. In the worst case, each cluster is non-redundant and we have to compare the ith cluster against $i-1$ many clusters resulting in $\sum_{i=1}^{|Clusters|} i-1 = \frac{(|Clusters|-1)\cdot(|Clusters|-2)}{2}$ many comparisons. Since the result set, however, is usually much smaller than the number of clusters, the bound $|Clusters| \cdot |Result|$ is often more precise. Thus, the overall complexity is given by the following theorem:

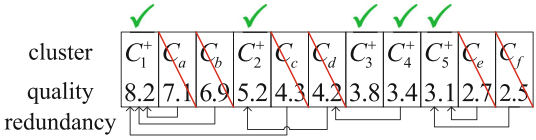


Fig. 3 Ranking of clusters to determine the optimal clustering

Theorem 3 *Given a set of twofold clusters $Clusters$, the optimal clustering $Result$ according to Def. 6 can be determined in time*

$$\mathcal{O}\left(|Clusters| \cdot log\left(|Clusters|\right) + |Clusters| \cdot |Result|\right) = \mathcal{O}\left(|Clusters|^2\right)$$

Since computing the optimal result is polynomial with respect to the number of clusters, however, #P-hard with respect to the number of vertices, the number of clusters has to depend superpolynomially on the number of vertices. Otherwise, i.e. assuming a polynomial bound, the overall complexity would also be polynomial. Thus, the number of clusters can be large in general and highlights the need for removing redundant clusters as proposed by our model.

Corollary 2 *Given a vertex-labeled graph $G = (V, E, l)$, the number of twofold clusters $Clusters$ cannot be polynomially bounded with respect to the number of vertices V.*

3.4 Discussion of parameters

With our GAMer model, we propose a flexible clustering approach, which can easily be adapted based on the users' needs. Even though a high flexibility is beneficial for the analysis, it may also hinder the model's application in practice since different parameters have to be set up. Thus, in the following, we recommend specific parameter settings, which can act as a starting point for an in-depth analysis of the data under consideration.

First, the user is able to adapt the quality assessment of the individual clusters (cf. Def. 4). If no specific preferences are given, we recommend the choice of $a = b = c = 1$. In this case, all characteristics are equally important, which leads a sound synthesis of subspace clustering and dense subgraph mining.

Second, the redundancy between different clusters can be controlled by r_{obj} and r_{dim} (cf. Def. 5). By selecting larger values, a higher overlap between the clusters is allowed. For many applications, however, we recommend the parameter setting $r_{obj}, r_{dim} \to 0$ since it leads to an easy interpretation of the clustering result: two clusters are either disjoint with respect to their objects or with respect to their dimensions. Thus, an object is not clustered twice within a single dimension. Please note that this result is still superior to projected clustering, which requires disjoint object sets.

Last, the user has to specify the characteristics that each individual cluster has to fulfill (cf. Def. 3). Actually, the parameters s_{min}, γ_{min}, and n_{min} do not influence the characteristics of the clusters, but they simply control the number of valid clusters. That is, by lowering these thresholds, the set $Clusters$ gets larger. Indeed, the overall clustering is quite robust if the threshold values are selected sufficiently small. For small thresholds, the set $Clusters$ would contain – besides the most interesting clusters – also clusters with low quality values. These uninteresting clusters, however, would mostly be excluded from the clustering result due to our redundancy model. Thus, if no further knowledge is given, one could simply ignore the thresholds. Please note, however, that as a consequence the runtime of the algorithm might increase since more clusters need to be analyzed. Furthermore, often only quasi-cliques with a density of at least 0.5 are considered interesting, as they are connected "tightly and relatively evenly" (cf. [45]). Thus, one should restrict the set of valid clusters a-priori by the three thresholds (e.g. $\gamma_{min} = 0.5$, $n_{min} = 4$, $s_{min} = 2$) to achieve far lower runtimes of the overall algorithm. Overall, only the parameter w is important for the clusters' characteristics, and it is easy to set since it simply controls the maximal extent of a cluster in the attribute space. Furthermore, heuristics to select the parameter w based on the given data are discussed in [32,43].

In summary, the proposed default parameters can be used to get a first understanding of the data's clustering structure. Moreover, for an in-depth analysis, our model offers the necessary flexibility by varying the individual properties.

4 The GAMer Algorithm

In the following section, we introduce our algorithm GAMer for efficiently determining the optimal twofold clustering. The algorithm interweaves the processes of calculating twofold clusters and of selecting the optimal clustering among these. By utilizing model-specific properties based on the cluster and clustering definition, we are able to exclude those vertex sets from our considerations that cannot be valid clusters or that might be valid clusters but will not be selected for the final result anyway because of the redundancy criteria. Through this early pruning we reduce the amount of analyzed vertex sets. Furthermore, our interweaved execution of both processes is preferable over a sequential one because it further increases the pruning potential.

4.1 Pruning based on cluster definition

In a naive approach we would have to check $2^{|V|}$ many subsets $O \subseteq V$ whether they fulfill our twofold cluster definition. Instead, we use Definition 3 systematically for the early pruning of vertex sets that cannot lead to valid clusters. The combination of our two properties, subspace cluster and quasi-clique, is crucial.

4.1.1 Initial pruning

Based on our subspace cluster property, two neighboring vertices can only belong to the same cluster if they are similar (i.e. their variation is smaller than w) in at least s_{min} dimensions. Thus, we remove all edges of our graph whose adjacent vertices do not fulfill this property. These edges cannot contribute to *any* cluster. We just retain the edges

$$E_{new} = \{(u, v) \in E \mid \exists S \subseteq Dim : |S| \geq s_{min} \wedge \forall d \in S : |u[d] - v[d]| \leq w\}.$$

Furthermore, our twofold clusters have to reach a minimal density and a minimal size. Thus, each vertex has to exceed the degree of $\lceil \gamma_{min} \cdot (n_{min} - 1) \rceil$ (after removing the edges in the previous step) to be a potential cluster object. Vertices that do not fulfill this property are removed, i.e. we just retain vertices with

$$deg^V(v) \geq \lceil \gamma_{min} \cdot (n_{min} - 1) \rceil .$$

By removing vertices, the degrees of other vertices can decrease accordingly. Thus, we iteratively check the minimal degree of the vertices until no more vertices can be removed. By this initial pruning our graph gets more sparse and might be even decomposed in several connected components. Since a twofold cluster must be connected, each component can be processed separately, which is more efficient than handling the original graph.

4.1.2 Efficient enumeration of twofold clusters

To enumerate the vertex sets in our graph, we use the set enumeration tree [34]. A complete tree for a graph with four vertices is depicted in Fig. 4. Each node of the tree represents a set

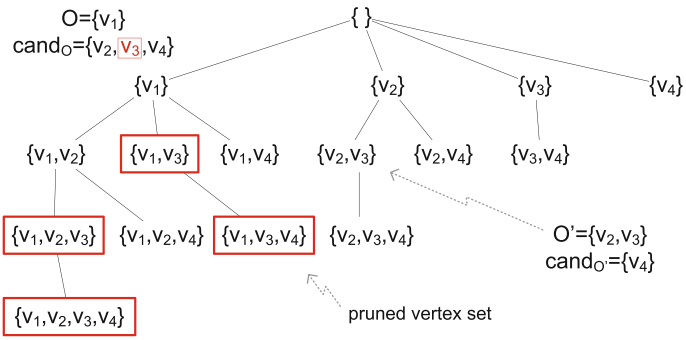


Fig. 4 Exemplary set enumeration tree and pruned vertex sets

of vertices $O \subseteq V$. All nodes of the tree together represent the powerset of V, which is the set of all potential clusters. On level i of the tree the cardinality of the sets is i. To build the tree a total order $\prec$ of the vertices is needed. In Fig. 4, we use the order $v_1 \prec v_2 \prec v_3 \prec v_4$. Each node O is associated with a candidate set $cand_O$, which contains all vertices that are ordered behind the vertices in O ($cand_O = \{v_i \in V \mid \forall v_j \in O : v_j \prec v_i\}$). A child node O' extends its parent node O through one of the vertices in $cand_O$. Thus, the subtree of a node O represents all potential node sets X such that $O \subset X \subseteq O \cup cand_O$. Please note that it is possible to use different orderings of vertices in each tree node O to sort the vertices from $cand_O$. This property will be beneficial for the overall algorithm in Sect. 4.3.

By pruning the set enumeration tree we narrow down the search space of vertex sets that we have to check against our cluster property. Note that the quasi-clique property is not monotone [31], and thus, we cannot simply prune a whole subtree if the parent node is not a valid cluster. Instead we prune a vertex v from the candidate set of a node O, if $\{v\} \cup O$ could never result in a valid cluster, not even by adding further vertices. If we were able to remove e.g. the vertex v_3 from the set $cand_{\{v_1\}}$, the highlighted $2^{|cand(O)|-1}$ many subsets in Fig. 4 would disqualify themselves as clusters without further analysis.

The vertices to be pruned can be inferred from our cluster definition. In contrast to the initial pruning step where the subspace cluster property and the quasi-clique property are used *separately* for pruning, we now integrate both paradigms in *combined pruning techniques* to achieve synergetic effects and thus a higher efficiency gain. We use the pruning techniques described in the following sections to avoid generating invalid clusters.

4.1.3 Pruning by subspace diameter

In [31] it is shown that the diameter of a γ_{min}-quasi-clique is restricted by an upper bound $k(\gamma_{min})$. Thus, each quasi-clique X containing the vertex t has to be a subset of t's k-neighborhood $N_k^V(t)$, i.e. $X \subseteq N_k^V(t)$. The neighborhood is given by:

$$N_k^V(t) = \{v \in V \mid d_V(t, v) \leq k\}$$

Here, $d_V(t, v)$ is the number of edges in the shortest path between t and v using only vertices from V.

Considering a vertex set O and a candidate set $cand_O$, we therefore can delete all vertices from $cand_O$ that are not contained in $\bigcap_{u \in O} N_k^V(u)$, since by adding one of these vertices to O we would lose the quasi-clique property. Note that our vertex sets O incrementally

grow by adding single vertices according to the set enumeration tree. Thus, the current set O is obtained by a step $O = O' \cup \{t\}$ with $t \in cand_{O'}$ where $cand_{O'}$ is known. Since $cand_O \subseteq cand'_O$ has to hold, we can incrementally determine $cand_O$ by just checking the neighborhood of the 'novel' vertex t, i.e. $cand_O = \{v \in cand_{O'} \mid t \prec v \wedge v \in N_k^V(t)\}$.

If we were only searching for quasi-cliques, we could just use the edge information for pruning (like in the Quick algorithm [21]). However, as our clusters also have to fulfill the subspace cluster property, our pruning technique additionally uses the attributes of the vertices for excluding unpromising vertices. Note that at this step in the algorithm the subspace $S(O)$ is already known and it holds that $|S(O)| \geq s_{min}$ (otherwise we would not have to process this subtree as it would not contain any valid cluster). To form a valid cluster, a set of vertices has to be a subspace cluster in at least s_{min} dimensions. Every vertex v that could be an element of a cluster $C = (X, S)$ with $X \supset O$ has to 'fit' into the subspace $S(O)$ in at least s_{min} many dimensions, i.e. the distance between v and any vertex from O may not be higher than w in the corresponding dimensions.

To use this fact for pruning we restrict the k-neighborhood of the vertex $t \in O$ to the vertices that also 'fit' into the subspace of O:

$$N_k^V(t, O) = \{x \in N_k^V(t) \mid |\{d \in S(O) \mid \forall u \in O : |u[d] - x[d]| \leq w\}| \geq s_{min}\}$$

We can now prune the candidate set using the new k-neighborhood, i.e. the set $cand_O$ contains just the vertices $\{v \in cand_{O'} \mid t \prec v \wedge v \in N_k^V(t, O)\}$.

The concept of this pruning technique is depicted in Fig. 5. Let us assume the parameters are set to $s_{min} = 2$, $w = 1$ and the maximal diameter be computed as $k(\gamma_{min}) = 2$. Thus, the current subspace of the set O is given by $S(O) = \{1, 2\}$. Using the traditional pruning technique, i.e. just considering the graph information, we are only able to prune the vertex y since its shortest path to t is longer than $k = 2$. Following our new definition we see that also u can be pruned. Although u is similar to t in 2 dimensions, it is only similar in $1 < s_{min}$ dimension with respect to the current subspace $S(O)$ since dimension 3 is currently not relevant. Thus, if we added u to O, we would violate the subspace cluster property. We can safely prune the vertex u from $cand_O$. Accordingly, our novel procedure only retains the vertices x and w as candidates.

Considering vertex x we can further enhance the effectiveness of our pruning technique. Vertex x is part of the 2-neighborhood of t as there exists a path of length 2 between t and x (path $t - u - x$). However, since the vertex u is pruned from the candidate set, the path $t - u - x$ is not valid when restricting to vertices of the candidate set; thus, the vertex x

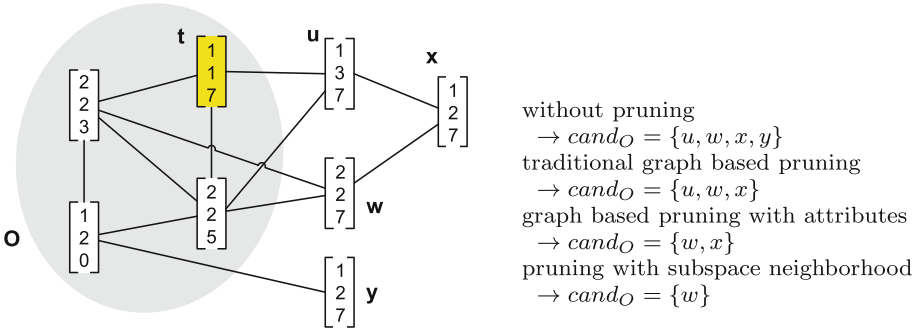


Fig. 5 Pruning of vertices based on diameter and neighborhood calculations

also has to be excluded as the new shortest path between t and x has length 3. Thus, for the neighborhood/shortest path computation it is not allowed to use the whole set V but only the non-pruned vertices. Formally, we recursively have to determine the neighborhood

$$NonPruned_{i+1} = N_k^{NonPruned_i}(t, O) \text{ starting with } NonPruned_0 = V$$

until a fixpoint is reached, i.e. $NonPruned_i = NonPruned_{i+1}$ holds.

Definition 7 (*Subspace k-neighborhood*) The subspace k-neighborhood $SN_k^V(t, O)$ of a vertex $t \in O$ is the largest set $NonPruned \subseteq V$ with $N_k^{NonPruned}(t, O) = NonPruned$.

Overall, the set $cand_O$ contains just those vertices of $cand_{O'}$ that are also located in t's subspace k-neighborhood, i.e. $cand_O = \{v \in cand_{O'} \mid t \prec v \wedge v \in SN_k^V(t, O)\}$. In our algorithm we efficiently determine this subspace neighborhood by a (restricted) depth-first search in the graph. Starting the depth-first search in the vertex t we can stop the traversal at a vertex v if either the path is longer than k or the subspace property is violated. Thus, in our example the traversal in the vertex u does not continue, we do not reach the vertex x in 2 steps and x is finally pruned. Overall our pruning technique integrating subspace properties prunes all vertices except of w, which shows the superiority over the traditional graph based approach.

4.1.4 Pruning by subspace vertex degrees

Our algorithm uses several pruning techniques based on the vertex degrees. In traditional quasi-clique mining [21] these pruning methods just consider graph information, while our novel pruning technique again integrates edge and attribute information to achieve higher pruning effectiveness. To apply the pruning techniques, we generally have to distinguish between the "indegree" $indeg$ and the "exdegree" $exdeg$ of a vertex u with respect to the sets O and $cand_O$. In traditional quasi-clique mining these two definitions are given as

$$indeg^O(u) = |\{x \in O | (u, x) \in E\}|, exdeg^O(u) = |\{x \in cand_O | (u, x) \in E\}|$$

Considering Fig. 6 we get $indeg^O(u) = 4$ and $exdeg^O(u) = 3$. The indegree describes the number of adjacent vertices of u in the set O (marked in red), and the exdegree describes the number of adjacent vertices in the candidate set $cand_O$ (marked in blue). Both values are used to estimate the possible size and density of quasi-cliques that contain the vertex u. Considering the vertex t, we get $indeg^O(t) = 1$ and $exdeg^O(t) = 2$.

Using the notion of indegree and exdegree, the methods of [21,46] introduce different pruning techniques. In general, the smaller the indegree/exdegree values of a vertex, the higher the chance to prune this vertex. Thus, our goal is to derive more accurate estimations for the indegree/exdegree values, i.e. smaller values should be determined by considering the special characteristics of our twofold clusters.

Degree values with respect to subspaces The basic idea of our pruning is introduced by the example in Fig. 6. Considering the traditional – just graph based – definitions for the indegree and exdegree we get in Fig. 6 the values $indeg^O(u) = 4$ and $exdeg^O(u) = 3$. However, in our model the vertices of a cluster do not only have to fulfill the quasi-clique property but also the subspace cluster property. Let us assume the parameters with respect to the subspace cluster property are set to $s_{min} = 2$ and $w = 1$. We see that x can only be added to the set O if we just consider subsets of the subspace $S = \{1, 3\}$. If we would select the second dimension, e.g. the subspace $\{1, 2\}$, the subspace property would be violated by

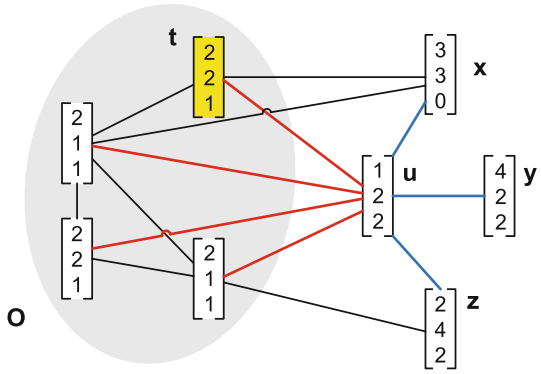


Fig. 6 Example for novel indegree and exdegree definitions based on subspaces

$O \cup \{x\}$. Thus, intuitively the indegree for the vertex x is zero or undefined for subspaces including dimension 2.

Considering the traditional exdegree, x contributes to the exdegree of u since both vertices are adjacent. However, examining e.g. the subspace $\{1, 2\}$, we know that $O \cup \{x\}$ cannot lead to a valid cluster. Thus, x should not contribute to the exdegree of u in this subspace. If we use the subspace $\{1, 3\}$, x can be added to O. However, in this subspace, u and x are not similar. They cannot both belong to a valid subspace cluster, and hence, u's exdegree should not include x. Generally, an edge between vertices u, x should only be considered if *both* vertices *together* can belong to a subspace cluster $C = (X, S)$ with $O \cup \{u, x\} \subseteq X \subseteq O \cup cand_O$ and $|S| \geq s_{min}$. In our example, x should not contribute to the exdegree of u at all.

In contrast, the vertices y and u can be added to O together; y should contribute to the exdegree of u. Similarly, z and u can belong to a valid subspace cluster; z should also contribute to u's exdegree. However, since $\{y\} \cup O$ leads to a subspace cluster in the subspace $\{2, 3\}$ and $\{z\} \cup O$ in $\{1, 3\}$, it would not make sense if both vertices contributed to the exdegree of u at the same time. Of course, if we look at the subspace $\{3\}$, both vertices contribute to u's exdegree; however, $\{3\}$ is not a valid subspace since $s_{min} = 2$.

Abstracting from this example, we see that the currently considered subspace S determines the vertices which contribute to the degree of u. Thus, we can determine for each dimension $d \in S(O)$ an individual indegree and exdegree which provides an upper bound for the corresponding value for any subspace containing the dimension d.

Definition 8 (*Dimension-wise indegree and exdegree*) The indegree and exdegree of a vertex u with respect to the vertex set O, the candidate set $cand_O$, and the dimension d are

$$indeg^O(u,d) = \begin{cases} |\{x \in O \mid (u,x) \in E\}| & \text{if } u[d] \in [upper[d] - w, lower[d] + w] \\ -1 & \text{else} \end{cases}$$

$$exdeg^O(u,d) = \begin{cases} -1 & \text{if } indeg^O(u,d) = -1 \\ |\{x \in cand_O \mid (u,x) \in E \wedge indeg^O(x,d) \neq -1 & \\ \wedge |u[d] - x[d]| \leq w\}| & \text{else} \end{cases}$$

with $upper[d] = \max_{o \in O}\{o[d]\}$ and $lower[d] = \min_{o \in O}\{o[d]\}$.

By using the values $upper[d]$ and $lower[d]$, we can easily check whether the adding of a vertex u to O violates the subspace property. In Fig. 6, for example, we get $upper[d_1] = 2$ and $lower[d_1] = 2$. Thus, any vertex with an attribute value between $[2 - 1, 2 + 1] = [1, 3]$ in this dimension can potentially lead to a cluster. If the attribute value of a vertex u is not

in this range, we cannot get a valid cluster. Thus, in this case we set $indeg^O(u, S) = -1$ to indicate that u cannot belong to a cluster in this subspace. Consequently, we are also able to set the vertex's exdegree to -1 if the indegree equals -1.

If a vertex $x \in cand_O$ should contribute to u's exdegree in the dimension d, it has to fulfill three requirements: First, it has to be adjacent to u. Second, its indegree is not undefined ($\neq -1$) since otherwise x cannot belong to the subspace S. Last, the vertices need to have similar attribute values in dimension d since otherwise they cannot belong to the same cluster.

Besides including subspace characteristics, one further advantage of our definition is the dependency between the exdegree and indegree values. While traditionally both values are computed independently, in our method the exdegree can be substantially lowered by taking the indegrees of the neighboring vertices into account.

Deriving single bounds for pruning To apply the pruning techniques introduced in [21,46] we need a single indegree/exdegree value per object. The smaller the indegree/exdegree value, the higher the chance to prune a vertex. Though, we have to guarantee that vertices belonging to valid clusters are not pruned; thus, we have to perform an optimistic estimation for the overall degrees of a vertex u: As the subspace of a valid cluster has to contain at least s_{min} dimensions, we choose the s_{min}-highest value over all dimensions $d \in S(O)$ as the indegree or exdegree of the vertex:

Definition 9 (*Subspace indegree and exdegree*) The subspace indegree and subspace exdegree of a vertex u with respect to the vertex set O and the candidate set $cand_O$ are defined as:

$$indeg^O_{new}(u) = s_{min}\text{-highest value from list } [indeg^O(u, d) \mid d \in S(O)]$$
$$exdeg^O_{new}(u) = s_{min}\text{-highest value from list } [exdeg^O(u, d) \mid d \in S(O)]$$

For vertex u in our example we would get $indeg^O_{new}(u) = 4$ as the attribute values of u fit into the range of O in any dimension. For the exdegree of the vertex u in the first dimension, we can only count vertex z; thus, we get $exdeg^O(u, d_1) = 1$. In the second dimension y contributes to u: we get $exdeg^O(u, d_2) = 1$. In the third dimension both vertices increase the exdegree of u, i.e. $exdeg^O(u, d_3) = 2$. Since each valid subspace has to cover at least $s_{min} = 2$ dimensions, we select the second highest value and we get an overall value of $exdeg^O_{new}(u) = 1$. This value is considerably smaller than the traditional exdegree of 3. Even by selecting $s_{min} = 1$, our pruning yields a tighter estimation; we then get an exdegree of 2. As our indegree and exdegree definitions just add further restrictions to the original definitions, the following inequalities hold:

Corollary 3 (Bounds for the indegree and exdegree values) *Given a set O and the corresponding candidate set $cand_O$, the following bounds hold for each $u \in O \cup cand_O$:*

$$indeg^O_{new}(u) \leq indeg^O(u),\ exdeg^O_{new}(u) \leq exdeg^O(u)$$

The inequalities ensure that our novel pruning method yields tighter estimations of the degrees than just using graph based pruning. Nevertheless, we realize an efficient computation of the degrees. Overall, this contributes to a more efficient cluster computation as we can prune more vertices from the candidate set without losing valid clusters.

4.1.5 Summary

In this section, we proposed several pruning techniques that are based on our cluster definition. Our pruning techniques jointly exploit the attribute data of the vertices and the graph data

for pruning. We have shown that by taking both data types into account we are often able to obtain smaller candidate sets and thus a higher pruning effectiveness. The introduced pruning techniques enable a quick identification of subtrees in the set enumeration tree that cannot lead to valid clusters. In GAMer we (mostly) traverse the set enumeration tree in a depth-first manner, we check the cluster properties for the current vertex set, and we prune subtrees with our methods above. Thus, we can efficiently generate all twofold clusters.

4.2 Pruning based on clustering definition

Since our clustering model does not allow any redundant clusters in the result set, it is worthwhile to early prune whole sets of (potential) clusters that would not be selected for the result set anyway – already before checking their validity. Through the beforehand introduced pruning, we avoid analyzing *invalid clusters*; in the following we additionally avoid generating *valid ones* that are redundant and thus are not allowed for the result.

4.2.1 Pruning of cluster collections

The following pruning methods utilize our clustering definition or redundancy model respectively. Please keep in mind that our redundancy relation is not transitive; thus, we cannot easily discard redundant clusters. As mentioned in Sect. 3.3.2, we can use a ranking of clusters to enable the efficient determination of our optimal clustering (cf. Fig. 3). We use this ranking idea for our algorithm. In the set enumeration tree, however, we do not want to check each node individually for its validity. Thus, we represent sets of *not yet analyzed nodes* by so-called cluster collections that are also included in our ranking. Each cluster collection represents a subtree in our set enumeration tree, i.e. a set of potential clusters. The basic idea of our method is to exclude those cluster collections $Coll_i$ from validity considerations for which a cluster C_r exists such that *all* clusters represented by $Coll_i$ are redundant with respect to the corresponding C_r. Thus, if C_r is added to $Result$, we can immediately remove the whole collection – and thus the whole subtree – from our queue. This idea is illustrated in Fig. 7. The cluster collection $Coll_A$ represents three clusters (not yet knowing whether they are valid ones) and each of these possible sets is redundant to the cluster C_1^+. If we select C_1^+ for the result we can directly remove $Coll_A$ and thus a whole set of not yet analyzed clusters.

Definition 10 formalizes the stored information within a cluster collection and shows the link to the set enumeration tree. Based on an already analyzed node $O \subseteq V$ in the set enumeration tree and its (pruned) candidate set $cand_O$, a cluster collection represents all nodes X with $O \subset X \subseteq O \cup cand_O$. Thus, a cluster collection represents all potential clusters within the subtree of O.

Definition 10 (*Cluster collection*) A cluster collection $Coll = (O, cand_O, S(O), Q_{max}, C_r)$ consists of:

– A set of vertices $O \subseteq V$ and the (pruned) candidate set $cand_O$

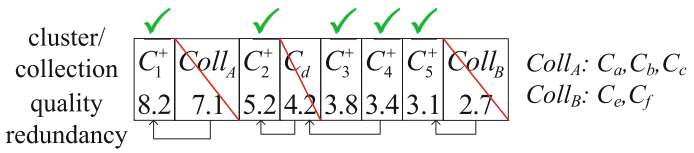


Fig. 7 Extended ranking that considers clusters and cluster collections

– The relevant dimensions $S(O)$ of the subspace cluster of O
– An upper bound Q_{max} for the quality of all represented clusters, i.e.
  $\forall X: O \subset X \subseteq O \cup cand_O$ it holds: X induces a valid cluster $C \Rightarrow Q_{max} \geq Q(C)$
– An anchor cluster C_r, to which all represented clusters are redundant, i.e.
  $\forall X: O \subset X \subseteq O \cup cand_O$ it holds: X induces a valid cluster $C \Rightarrow C \prec_{red} C_r$

The maximal quality Q_{max} is required for inserting the cluster collection at the correct position within the ranking. Furthermore, we need this quality as well as the subspace S and the vertex sets to calculate whether the cluster collection is redundant with respect to other clusters. In the remainder of this subsection, we will provide bounds for the maximal possible overlap between the objects/dimensions of the represented clusters and the ones of C_r, and we derive bounds for the maximal possible quality of the represented clusters. Based on these bounds we can ensure that any vertex set X with $O \subset X \subseteq (O \cup cand_O)$ will be redundant to C_r. Thus, by pruning the cluster collection, we do not wrongly discard some valid clusters.

A cluster collection represents a whole set of potential clusters without actually generating them. If we discard such a collection we get a high efficiency gain. During the enumeration of our clusters, i.e. while traversing the set enumeration tree, we use two different techniques to generate such cluster collections and their corresponding anchor clusters C_r.

Superset collection The first technique is illustrated on the left hand side of Fig. 8. The aim of this technique is to prevent the processing of a subtree rooted by the set O if $C_O = (O, S(O))$ is a valid cluster and every possible cluster in the subtree would be redundant with respect to the cluster C_O. We aim at excluding redundant *supersets* of the anchor cluster C_O. This constellation occurs especially when the quality function prefers clusters with high density or dimensionality.

To get a valid superset collection $Coll_{sup} = (O, cand_O, S(O), Q_{max}, C_O)$, we have to ensure that each represented cluster $C = (X, S)$ with $O \subset X \subseteq (O \cup cand_O)$ is redundant with respect to C_O. In the following, we introduce properties a superset collection has to fulfill to guarantee the validity of $C \prec_{red} C_O$, i.e. the three conditions from Definition 5 are true for any cluster $C = (X, S)$ in this case. First, we assume that upper bounds for the size, the dimensionality and the density of such clusters $C = (X, S)$ are given by n_{max}, s_{max} and γ_{max}.

1. *Lower quality:* If $Q_{max} < Q(C_O)$ is true, then $Q(C) < Q(C_O)$ holds for each represented C. By choosing $Q_{max} = \gamma_{max}^a \cdot n_{max}^b \cdot s_{max}^c$ this conclusion is obviously valid.

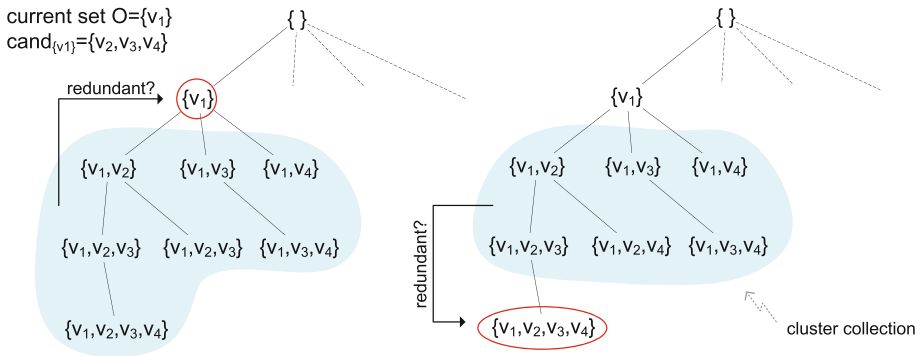


Fig. 8 Two types of cluster collections: superset collection (*left*) and subset collection (*right*)

2. *Object overlap:* If $n_{max} \leq \frac{|O|}{r_{obj}}$ is true, then $\frac{|X \cap O|}{|X|} \geq r_{obj}$ holds for each represented C. Since $O \subset X$, we get $\frac{|X \cap O|}{|X|} = \frac{|O|}{|X|} \geq \frac{|O|}{n_{max}} \geq \frac{|O| \cdot r_{obj}}{|O|} = r_{obj}$.
3. *Dimension overlap:* In any case $\frac{|S \cap S(O)|}{|S|} \geq r_{dim}$ holds for each represented C. Because of the antimonotonicity of the subspace property for every cluster C with $O \subset X$ it holds that $S \subseteq S(O)$. Thus, $\frac{|S \cap S(O)|}{|S|} = 1 \geq r_{dim}$.

If the superset collection fulfills these properties for the redundancy, we store the cluster C_O and the cluster collection $Coll_{sup}$ in the queue and we stop traversing the subtree rooted by O. An additional benefit of this technique is that if we get $n_{max} < n_{min}$ or $\gamma_{max} < \gamma_{min}$ we know that we cannot find a valid cluster at all in this subtree. In this case, we do not have to search the subtree in any case, so we do not store the cluster collection and just stop processing the subtree.

In the following we determine the upper bounds for the size, dimensionality and density of the represented clusters C.

Theorem 4 (Upper bounds for superset collection) *For every $C = (X, S)$ with $O \subset X \subseteq O \cup cand_O$ the following bounds apply:*

a) $|S| \leq |S(O)| =: s_{max}$

b) $\gamma(X) \leq \frac{min\_deg}{|O|} =: \gamma_{max}$ *with* $min\_deg = \min_{v \in O}(indeg^O_{new}(v) + exdeg^O_{new}(v))$

c) $|X| \leq \min(\lfloor \frac{min\_deg}{\gamma_{min}} \rfloor + 1, |O \cup cand_O|) =: n_{max}$

Proof a) Because of the antimonotonicity of the subspace cluster property it holds for every cluster $C = (X, S)$ with $X \supset O$: $S \subseteq S(O)$. Thus, we get $|S| \leq |S(O)|$.

b) The maximal density of a cluster $C = (X, S)$ is determined by the minimal degree of the vertices of X:

$$\min_{v \in X}(deg^X(v)) \leq \min_{v \in O}(indeg^O_{new}(v) + exdeg^O_{new}(v)) = min\_deg$$

We only consider vertices from O as the vertices from $cand_O$ do not have to be part of X, and thus, a vertex from $cand_O$ with a small degree does not necessarily influence the minimal degree of the vertices of X.
Thus, for the density of the vertex set X it holds that

$$\gamma(X) = \frac{\min_{v \in X}(deg^X(v))}{|X| - 1} \leq \frac{min\_deg}{|X| - 1}$$

As X is a superset of O it holds that $|X| \geq |O| + 1$ and thus

$$\gamma(X) \leq \frac{min\_deg}{|O|} =: \gamma_{max}.$$

c) The maximal size n_{max} of a cluster $C = (X, S)$ is also determined by the minimal degree min_deg. Every valid cluster has to fulfill

$$min\_deg \geq \min_{v \in X}(deg^X(v)) \geq \lceil \gamma_{min} \cdot (|X| - 1) \rceil \Leftrightarrow \lfloor \frac{min\_deg}{\gamma_{min}} \rfloor \geq |X| - 1$$

Furthermore, the size of a cluster $C = (X, S)$ with $X \subseteq (O \cup cand_O)$ cannot be larger than $|O \cup cand_O|$. Thus, the maximal size of a valid cluster is

$$n_{max} = \min\left(\lfloor \frac{min\_deg}{\gamma_{min}} \rfloor + 1, |O \cup cand_O|\right).$$

□

Subset collection The second pruning technique is illustrated on the right hand side of Fig. 8. Before a set O is extended by a single further vertex $v \in cand_O$, we check the validity of the cluster obtained by the set $O \cup cand_O$, i.e. by adding all vertices of $cand_O$ simultaneously. We denote this cluster by $C_r = (O_r, S_r)$ with $O_r = O \cup cand_O$ and $S_r = S(O \cup cand_O)$. If C_r is a valid cluster and if every possible cluster $C = (X, S)$ with $O \subset X \subset (O \cup cand_O)$ is redundant with respect to C_r we can again generate a cluster collection, store it in the queue and we can *stop processing this subtree*. In contrast to the previous technique we aim at excluding redundant *subsets* of the anchor cluster whereas in the first technique we excluded redundant supersets. Especially for quality functions that prefer large clusters, this is a likely constellation.

To get a valid subset collection $Coll_{sub} = (O, cand_O, S(O), Q_{max}, C_r)$, we have to ensure the redundancy of each represented cluster with respect to C_r. Thus, for every possible cluster $C = (X, S)$ with $O \subset X \subset (O \cup cand_O) = O_r$ the three conditions from Definition 5 must be true. If the subset collection fulfills certain properties we can guarantee the validity of these conditions. Again, we first assume upper bounds n_{max}, s_{max} and γ_{max} for the size, dimensionality and density of such clusters are given.

1. *Lower quality:* If $Q_{max} < Q(C_O)$ is true, then $Q(C) < Q(C_O)$ holds for each represented C. By choosing $Q_{max} = \gamma_{max}^a \cdot n_{max}^b \cdot s_{max}^c$ this conclusion is obviously valid.
2. *Object overlap:* In any case $\frac{|X \cap O_r|}{|X|} \geq r_{obj}$ holds for each represented C. As every possible X in this subtree is a subset of O_r, we get $\frac{|X \cap O_r|}{|X|} = 1 \geq r_{obj}$.
3. *Dimension overlap:* If $\frac{|S(O) \cap S_r|}{|S(O)|} \geq r_{dim}$ is true, then $\frac{|S \cap S_r|}{|S|} \geq r_{dim}$ holds for each represented C. Due to the antimonotonicity of the subspace cluster property, for every S it holds that $S_r \subseteq S \subseteq S(O)$. Thus, $\frac{|S \cap S_r|}{|S|} = \frac{|S_r|}{|S|} \geq \frac{|S_r|}{|S(O)|} = \frac{|S(O) \cap S_r|}{|S(O)|} \geq r_{dim}$.

If the subset collection fulfills these properties for the redundancy, we store $Coll_{sub}$ in the queue and stop processing this subtree. Also the cluster C_r is stored as a valid cluster in the ranking.

Theorem 5 (Upper bounds for subset collection) *For every $C = (X, S)$ with $O \subset X \subset (O \cup cand_O)$ the following bounds apply:*

a) $|S| \leq |S(O)| =: s_{max}$
b) $\gamma(X) \leq \frac{min\_deg}{|O|} =: \gamma_{max}$ *with* $min\_deg = \min_{v \in O}(indeg_{new}^O(v) + exdeg_{new}^O(v))$
c) $|X| \leq |O \cup cand_O| - 1 =: n_{max}$

Proof a) As in the proof of Theorem 4 a)
b) As in the proof of Theorem 4 b)
c) For every set $X \subset (O \cup cand_O)$ it holds that $|X| \leq |O \cup cand_O| - 1$.

□

4.2.2 Pruning of cluster collections with respect to non-anchor clusters

The previous techniques avoid the processing of a collection if all possible clusters in this subtree would be redundant with respect to a certain anchor cluster C_r. If the anchor cluster is selected for the result, we can prune the whole cluster collection. However, if C_r itself turns out to be redundant, the corresponding subtree still has to be processed. However, if C_r is redundant it is likely that there exists a similar cluster C^* in the final result set such that every cluster from the collection would be redundant with respect to C^*. In this case, we can

also avoid the processing of the collection. The detection of such a cluster C^* is the aim of our next pruning technique.

Before we process a cluster collection $Coll = (O, cand_O, S(O), Q_{max}, C_r)$, we search the current result for a cluster $C^* = (O^*, S^*)$ such that each cluster from the collection is redundant to C^*. For this, the three properties according to Definition 5 have to hold for each cluster $C = (X, S)$ with $O \subset X \subseteq O \cup cand_O$. Again, we introduce properties the cluster collection has to fulfill to guarantee the redundancy of the represented clusters with respect to C^*.

1. *Lower quality:* If $Q_{max} < Q(C^*)$ is true, then $Q(C) < Q(C^*)$ holds for each represented C. This is obvious due to the definition of Q_{max}.
2. *Object overlap:* It has to hold $\frac{|X \cap O^*|}{|X|} \geq r_{obj}$ for any given X. Thus, if we find a lower bound for the number of overlapping vertices and if this smallest number of overlapping vertices still results in redundancy with respect to the objects, any cluster of the collection has to be redundant with respect to the objects. We show:

Theorem 6 *If*

$$\frac{|O^* \cap O| + \max\{n_{min} - |O| - |cand_O \backslash O^*|\,, 0\}}{\max\{n_{min}\,, |O| + |cand_O \backslash O^*|\}} \geq r_{obj}$$

is true, then $\frac{|X \cap O^*|}{|X|} \geq r_{obj}$ *holds for each represented cluster* $C = (X, S)$.

Proof To get a lower bound for the number of overlapping vertices $|X \cap O^*|$ we assume that the set X contains as few vertices from O^* as possible and as many vertices from $cand_O \backslash O^*$ as possible. As $X \supset O$, it also holds that $(X \cap O^*) \supset (O \cap O^*)$, and thus, the overlap $X \cap O^*$ has to contain at least $|O^* \cap O|$ many vertices. To get the minimum value for $\frac{|X \cap O^*|}{|X|}$, we first add all vertices from $cand_O \backslash O^*$ to X. However, if $|O| + |cand_O \backslash O^*| < n_{min}$ we have to add $n_{min} - |O| - |cand_O \backslash O^*|$ many vertices from O^* to X in order to get a valid cluster, by which the overlap increases. Thus, the minimal value for $\frac{|X \cap O^*|}{|X|}$ is

$$\frac{|O^* \cap O| + \max\{n_{min} - |O| - |cand_O \backslash O^*|\,, 0\}}{\max\{n_{min}\,, |O| + |cand_O \backslash O^*|\}}$$

□

3. *Dimension overlap:* It has to hold $\frac{|S \cap S^*|}{|S|} \geq r_{dim}$. Thus, if we find a lower bound for the number of overlapping dimensions and if this smallest number of overlapping dimensions still results in redundancy with respect to the dimensions, any cluster of the collection has to be redundant with respect to the dimensions. We show:

Theorem 7 *If*

$$\frac{s_{min} - |S(O) \backslash S^*|}{s_{min}} \geq r_{dim}$$

is true, then $\frac{|S \cap S^*|}{|S|} \geq r_{dim}$ *holds for each represented cluster* $C = (X, S)$.

Proof For the subspace S of any cluster from the collection it holds that $S \subseteq S(O)$ (anti-monotonicity). Furthermore, S has to fulfill the minimum dimensionality s_{min}. To get a lower bound for the number of overlapping relevant dimensions $|S \cap S^*|$ we assume that S covers as many dimensions from $S(O) \backslash S^*$ as possible. If $|S(O) \backslash S^*| < s_{min}$ we have to add

Algorithm 1: GAMer algorithm (main method)

```
1  Result = ∅ // current result set
2  queue = ∅ // priority queue containing clusters and collections,
                // sorted by their quality values in descending order
3  perform initial pruning // cf. Section 4.1.1
4  sort vertices v ∈ V descendingly by their vertex degree
5  for v ∈ V do
6      DFS_traversal({v}, V) // start one layer below the empty root
7      V = V\{v} // only vertices ordered behind v must be considered
8  while queue ≠ ∅ do
9      remove first object Obj from queue
10     if Obj is cluster then
11         // check redundancy
12         for C ∈ Result do
13             if( Obj ≺_red C) goto line 8 // discard redundant cluster
14         Result = Result ∪ {Obj} // cluster is non-redundant
15     else // Obj is collection Coll = (O, cand_O, S, Q_max, C_r)
16         if anchor cluster C_r ∈ Result then
17             goto line 8 // discard whole collection
18         // else check redundancy w.r.t. non-anchor clusters (cf. Sec. 4.2.2)
19         for C ∈ Result do
20             if Coll is redundant to C then
21                 goto line 8 // discard whole collection
22         // collection is non-redundant, restart traversal
23         sort vertices v ∈ cand_O descendingly by indeg^O_new(v)
24         for v ∈ cand_O do
25             DFS_traversal(O ∪ {v}, cand_O)
26             cand_O = cand_O\{v} // consider only vertices ordered behind v
27 return Result
```

$s_{min} - |S(O) \backslash S^*|$ many dimensions from the subspace S^* in order to get a valid cluster. The minimal overlap of the subspaces S and S^* can be computed as $s_{min} - |S(O) \backslash S^*|$. Thus, the minimal value for $\frac{|S \cap S^*|}{|S|}$ is $\frac{s_{min} - |S(O) \backslash S^*|}{s_{min}}$. □

If all three properties are fulfilled by the cluster collection, we can prune the collection $Coll$ because it is guaranteed that each of the represented clusters is redundant with respect to C^*.

4.2.3 Summary

In this section, we introduced several pruning techniques that are based on our clustering definition. While the pruning techniques proposed in the last section aimed at pruning subtrees of the set enumeration tree that cannot contain valid clusters at all, the techniques introduced here prune subtrees that only contain redundant clusters. By using these techniques we can avoid generating the set of all valid clusters and thus obtain a more efficient computation.

4.3 Overall algorithm

In our GAMer algorithm the method of the set enumeration tree, for generating the clusters, and the priority queue, for ranking the clusters, are nested. The pseudo codes are given in

Algorithms 1 and 2. After performing the initial pruning (line 3), we start the depth-first traversal of the set enumeration tree with the nodes containing just a single vertex (i.e. one layer below the empty root; line 6). During the depth-first traversal (function $DFStraversal$, Alg. 2) clusters and/or cluster collections are inserted into the queue. At the beginning, we prune the candidate set of the current node (line 28) according to the techniques introduced in Sect. 4.1. Thus, the search space is substantially narrowed. Furthermore, this step leads to compact cluster collections and thus to a more accurate estimation of their redundancy. If the current node represents a valid cluster C_O, we insert it into the queue (line 31), which is sorted according to the corresponding quality values. After this, we apply our pruning based on cluster collections: We generate the collections (line 32 and 41) and we check their redundancy with respect to the clusters C_O or C_r respectively (line 35 and 42). If one of these collections is redundant, we currently stop the traversal of this subtree, insert the collection into the queue and hopefully prune the whole subtree later on. If the collections are not redundant, we further have to descend into the subtrees and recursively invoke the traversal function for each vertex in the pruned candidate set (line 47). Note that the candidate set is sorted based on the indegree ($indeg^O_{new}(u)$) of the vertices. Thus, vertices potentially leading to clusters with high density are processed first, increasing the probability to generate the most interesting clusters first.

If the current depth-first traversal is finished (since either the collections are redundant or all vertices/clusters are processed) the algorithm returns to the queue processing step (lines 8 ff., Alg. 1). At each time we select the top ranked, i.e. highest quality, object from the queue (line 9). If it is a cluster, we check the redundancy with respect to clusters in the actual result set $Result$. If it is redundant, we discard the cluster (line 13) and proceed with the next object. If the cluster is non-redundant, we add it to the $Result$. If the object is a cluster collection $Coll = (O, cand_O, S, Q_{max}, C_r)$, we first check if the anchor cluster C_r is already included in the result (line 16). If so, we can safely prune the whole collection/subtree. If $C_r \notin Result$, we apply our pruning based on non-anchor clusters. Thus, the collection is potentially rejected. If even this pruning is not successful, we further have to traverse the set enumeration tree at the corresponding subtree (the previous stopping position; stored within the collection) as shown in lines 22–26. We refine this subtree, i.e. we generate further clusters and cluster collections that are inserted into the queue.

Because the queue is ranked based on the (estimated) quality values, we aim for the most interesting parts of the set enumeration tree. This way, in later steps we can discard many redundant cluster collections. We mainly generate clusters that are selected later on for the result and that are non-redundant. Overall, we use the subspace property, the quasi-clique property and the redundancy model simultaneously to achieve a speed-up of our algorithm.

5 Experiments

5.1 Experimental setup

We compare our GAMer method to CoPaM [26], the only method which also considers subspaces and dense subgraphs. To include a further competitor we extend Cocain [45] to handle our data. Originally, this method considers a set of graphs to find sets of vertices forming quasi-cliques in several of these graphs. Since we use attribute data, we generate one graph per dimension and retain only those edges of the original graph connecting vertices with similar attribute values in this dimension; thus, our Cocain° method simulates subspace clustering. Furthermore, we implement two baseline algorithms to analyze the efficiency of

Algorithm 2: GAMer algorithm ($DFS\_traversal$ method)

Input: current vertex set O, previous candidate set $cand_{O'}$

28 determine and prune $cand_O$ (based on $cand_{O'}$) // cf. Section 4.1
29 determine $C_O = (O, S(O))$ // current cluster
30 **if** C_O *is valid twofold cluster* **then**
31 insert C_O into $queue$
32 determine superset collection $Coll_{sup}$
33 **if** $n_{max} < n_{min}$ *or* $\gamma_{max} < \gamma_{min}$ **then**
34 **return;** // no further clusters in this subtree possible
35 **if** $Coll_{sup}$ *is redundant to* C_O **then**
36 insert $Coll_{sup}$ into $queue$
37 **return;** // (currently) stop DFS traversal
38 determine $C_r = (O \cup cand_O, S(O \cup cand_O))$ // look ahead
39 **if** C_r *is valid twofold cluster* **then**
40 insert C_r into $queue$
41 determine subset collection $Coll_{sub}$
42 **if** $Coll_{sub}$ *is redundant to* C_r **then**
43 insert $Coll_{sub}$ into $queue$
44 **return;** // (currently) stop DFS traversal
45 sort vertices $v \in cand_O$ descendingly by $indeg^O_{new}(v)$
46 **for** $v \in cand_O$ **do**
47 $DFS\_traversal(O \cup \{v\}, cand_O)$
48 $cand_O = cand_O \backslash \{v\}$ // consider only vertices ordered behind v

GAMer. These baseline algorithms generate the same clustering result as GAMer but do not simultaneously use the subspace and subgraph properties for generation and pruning of clusters. The first algorithm $SeqSubGraph$ starts by generating all quasi-cliques (based on [21]) and after this it checks the subspace property of these sets. The approach $SeqSubSpace$ starts by generating all subspace clusters (based on the Apriori principle) and afterward checks the quasi-clique property. At the end both algorithms remove the redundant clusters. Overall, these algorithms sequentially check the properties of twofold clusters. By default for our approaches, CoPaM and Cocain°, we use the parameter setting $\gamma_{min} = 0.5$, $s_{min} = 1$, $n_{min} = 10$. The redundancy parameters for GAMer are set to $r_{obj} = r_{dim} = 0.5$.

For our evaluation, we use synthetic data and several publicly available real world datasets. We provide all datasets and their descriptions as well as executables, parameter settings and evaluation measures on our website. We use *gene data* and their interactions obtained from [37,39] (3548 vertices; 8334 edges; 115 dimensions), *patent information*[1] (492007 vertices; 528333 edges; 5 dimensions), an extract of the *Arxiv database*[2] (13003 vertices; 120213 edges; 300 dimensions) and a co-author graph extracted from the *DBLP database*[3] (2482 vertices; 7438 edges; 11 dimensions). Furthermore, we generate synthetic data where the properties of the data can be specified. Our generator adapts the planted partitions model [6]. Intuitively, given the desired number of clusters and the vertices belonging to each cluster, we randomly add edges between and within the different clusters according to a specified density. Please note that we allow overlapping clusters, i.e. some vertices might belong to multiple clusters. The fraction of overlap can be controlled by the user. To generate the

[1] http://www.nber.org/patents/.

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

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

feature vectors, we specify the overall dimensionality of the data and we provide the number of relevant dimensions for each cluster. Given these information, we randomly select for each cluster a set a relevant dimensions. Please note that each cluster is associated with an individual set of relevant dimensions. By default we generate 20 dimensional data with 80 clusters, each with 15 vertices. 6 % of the clusters' vertices belong to more than one cluster, i.e. the data contains slightly overlapping clusters. The clusters' density is 0.6 and each cluster is associated with 5–10 relevant dimensions.

The efficiency of the approaches is measured by their runtime. For comparability all experiments were conducted on Opteron 2.3GHz CPUs and Java6 64 bit. For the clustering quality we calculate the F1 value, which is commonly used in evaluation of subspace clustering [12,28]. It is the harmonic mean of recall ("are all objects of the hidden cluster detected?") and precision ("how accurately is the cluster detected?") values, respectively. The F1 value of the entire clustering is the average of all clusters' F1 values.

5.2 Scalability on synthetic data

In the first experiment in Fig. 9a, we increase the database size by varying the number of clusters and keeping the number of objects per cluster fixed. Our GAMer algorithm is several orders of magnitude faster than the competing approaches (please note the logarithmic scale). Especially, the *SeqSubSpace* baseline method increases heavily and is not applicable on datasets with more than 1500 vertices. This approach generates a huge amount of subspace clusters, that, however, are not connected and hence are not valid clusters. The pruning in this approach is very limited. By incorporating the graph structure we can early reject vertex sets, which is exploited by our GAMer.

In the next experiment in Fig. 9b, we increase the database size by varying the number of objects per cluster and keeping the number of overall clusters fixed. This is even more challenging because by hiding larger clusters we consequently get larger candidate sets in the set enumeration tree and thus more nodes have to be analyzed. While all runtimes increase, our GAMer is still the most efficient approach. We see that our baseline algorithms, CoPaM, and Cocain° are already not applicable for small datasets. The various pruning strategies of GAMer enable the efficient generation of the clustering solution.

Figure 10 analyzes the effects when the data dimensionality is increased. The slopes of all curves are nearly identical. The high runtime of *SeqSubSpace* indicates that graph based pruning is more effective than subspace pruning. However, a combination of both is even

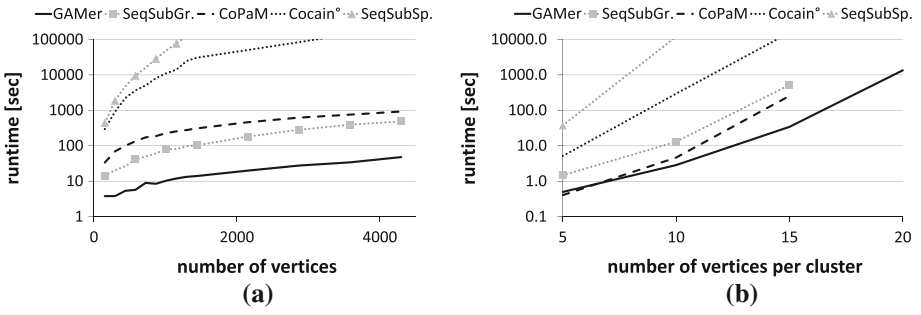


Fig. 9 Scalability of the algorithms with respect to the database size. **a** Varying number of clusters. **b** Varying number of vertices per cluster

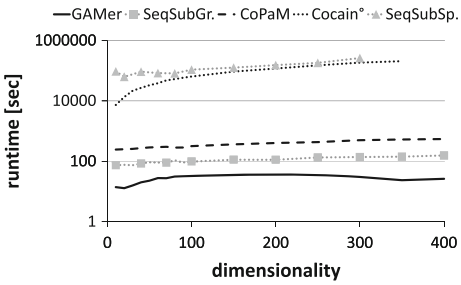


Fig. 10 Scalability of the algorithms with respect to the database dimensionality

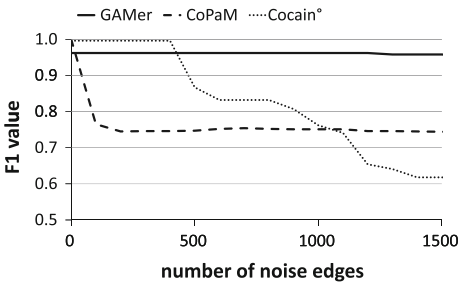


Fig. 11 Clustering quality versus noise

better because the absolute runtime of GAMer is still the lowest, as also indicated by the previous experiments.

5.3 Quality on synthetic data

In the following experiments, we exclude the baseline algorithms because the clustering results are identical to GAMer. In Fig. 11, we analyze the robustness of the algorithms with respect to noise. Besides the clustered objects, we add noise objects that do not belong to any cluster (25 % with respect to the former objects) and noise edges that connect vertices from different clusters. Our GAMer is nearly not influenced by noise and gets high quality. The qualities of CoPaM and Cocain° decrease. By adding noise, supersets of the hidden clusters can become dense. Because both maximize the number of vertices in a cluster, they misleadingly detect these supersets and include noise. GAMer, however, identifies the correct clusters since they have higher density and more relevant dimensions. Our trade-off between the three characteristics leads to better quality. By preferring maximal clusters with respect to their size, i.e. by setting the quality parameters to $a = c = 0, b = 1$, as the other models implicitly do, GAMer can also obtain perfect qualities in settings with low noise; however, by trading off the three characteristics ($a = b = c = 1$), we get high qualities for all settings.

In Fig. 12, we increase the degree of overlap, i.e. the overlapping vertices can belong to a maximal number of clusters that is depicted on the x axis (for the degree 1, the clusters do not overlap). Again, the high quality of GAMer is confirmed. GAMer can handle high overlap, because we focus on the most interesting and non-redundant clusters. CoPaM and Cocain° fail for these settings; the quality drops or the algorithms are not applicable at all due to extreme memory usage. Since CoPaM and Cocain° maximize the number of vertices, a high overlap leads to vertex sets combined of different clusters that are wrongly identified as clusters. Along with the redundancy models that do not exclude these clusters, low qualities

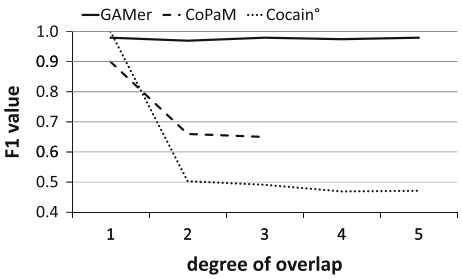


Fig. 12 Clustering quality versus overlap

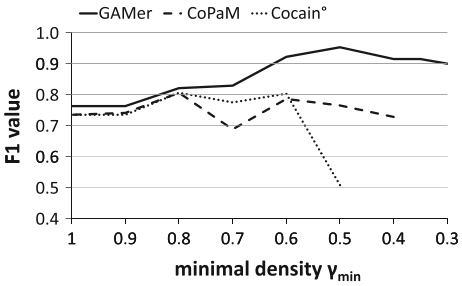


Fig. 13 Clustering quality versus varying minimal density γ_{min}

are obtained. Our redundancy model, in contrast, prevents to output uninteresting clusters that just represent the intersection of overlapping clusters.

Next, we analyze the effects of a varying minimal density γ_{min}. In Fig. 13, we generate clusters with densities between 0.5 and 0.8. At the beginning ($\gamma_{min} = 1$), none of the algorithms gets perfect quality. Since the hidden clusters have lower densities they cannot be detected with $\gamma_{min} = 1$; only parts of the clusters are identified. If the minimal density is decreased, the quality of GAMer increases. We detect more and more hidden clusters. For a sufficiently small γ_{min} the quality remains constantly high. Although several further subsets fulfill these minimal density, our redundancy model concentrates on the true ones. Cocain° shows a different behavior. Starting with a slight increase in clustering quality, it dramatically drops for low density values. For a low density threshold, many clusters are regarded as dense and these redundant clusters are not excluded from the result set. Based on similar reasons, CoPaM also shows poor clustering quality. Due to our redundancy model, GAMer is more robust with respect to the minimal density. Another drawback of CoPaM and Cocain° is that the minimal density has to be larger than 1/3 or 1/2 respectively. GAMer, however, can operate with arbitrary densities. Overall, all experiments indicate that GAMer achieves high clustering qualities by confining the result to the most interesting and non-redundant clusters.

5.4 Quality on real world data

For real world data, the ground truth is usually not given; determining the clustering quality is often not possible. For our gene data, however, we can use the Go-Miner tool [44] that assigns genes to biological categories. These classes are used as hidden clusters as also done in [26]. For this experiment in Fig. 14, GAMer obtains the highest quality results. The limited models of CoPaM and Cocain° are not able to detect meaningful clusters. Furthermore, we calculate for this experiment the results of approaches that consider only one paradigm, i.e. subgraph mining (maximal quasi-cliques, Quick [21]) or subspace clustering (Proclus [3]). The quality of these two algorithms is low, indicating that a synthesis of both paradigms – as

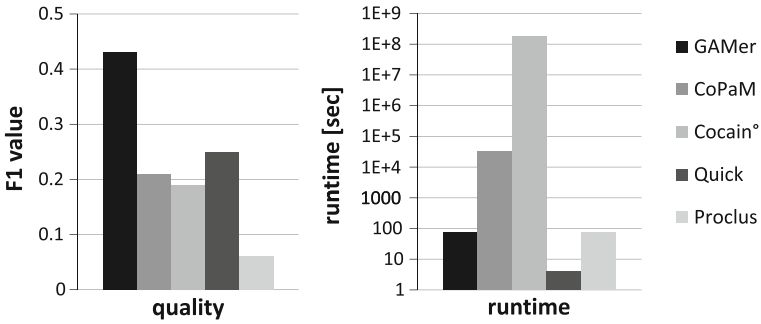


Fig. 14 Clustering quality and runtime of the algorithms for gene data

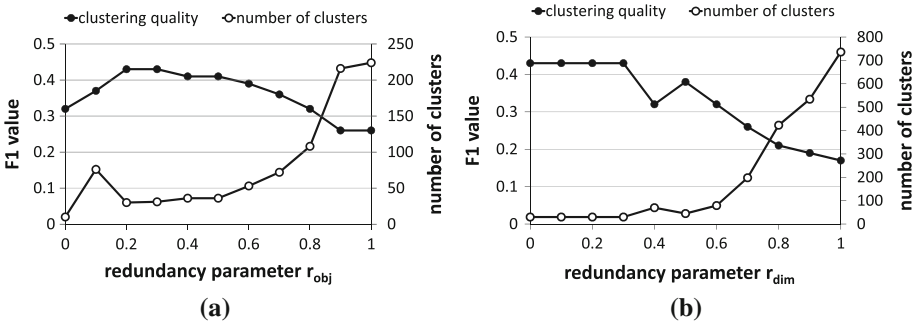


Fig. 15 Influence of the redundancy model on the clustering result (gene data). **a** Influence of parameter r_{obj}. **b** Influence of parameter r_{dim}

our model does – can effectively increase the clustering quality. Considering the runtime, we see that our approach is more than 100 times faster than CoPaM and even better compared to Cocain°.

Next, we analyze how the clustering quality is influenced by excluding redundant clusters for the previous dataset. In Fig. 15a we vary the redundancy parameter r_{obj}. A higher parameter assesses less clusters as redundant; thus, increasing the number of clusters in the result. The quality decreases accordingly because we include also low quality clusters in the result. However, it is interesting that too small values also result in low qualities. In this case, we exclude too many and also important clusters. In Fig. 15b, we analyze the influence of the parameter r_{dim}. As with r_{obj}, by increasing the parameter the overall clustering quality decreases. However, by choosing r_{dim} too small we get a very small result set, as many clusters are discarded as redundant. Overall, a modest removal of redundant clusters yields high-quality results.

For our remaining datasets (DBLP, Arxiv and the Patent data), we have no information about the hidden clusters and thus cannot compute F1 values. Thus, we analyze in Fig. 16 different properties of the clustering results determined by GAMer, CoPaM and Cocain°. The first observation is that the runtimes of CoPaM and Cocain° are orders of magnitude higher than the runtime of GAMer. For the Patent data CoPaM did not finish within 2 days; even worse Cocain° only finished on the DBLP data. Our approach is very efficient due to the developed pruning techniques. Considering the number of generated clusters, the huge result size of CoPaM becomes apparent. CoPaM excludes nearly no clusters; the redundancy

		Gene		Arxiv		DBLP			Patent
		GAMer	CoPam	GAMer	CoPam	GAMer	CoPam	Cocain°	GAMer
runtime [s]		76	33060	22	4440	6	22589	3963	574
number clusters		30	115581	23	98419	83	341333	151	146
cluster size	avg.	8.8	9.67	9.04	9.12	10.1	10.8	9.6	11.7
	std. dev.	1.16	1.54	0.71	1.38	1.04	1.48	0.97	2.54
density	avg.	0.72	0.24	0.64	0.23	0.62	0.2	0.62	0.6
	std. dev.	0.06	0.14	0.05	0.10	0.02	0.09	0.02	0.01
dimensionality	avg.	15.5	12.2	5.04	5.0	3.0	3.0	3.0	3.0
	std. dev.	5.10	4.00	0.21	0.03	0.0	0.01	0.0	0.0

Fig. 16 Clustering properties for different real world dataset (not listed methods did not finish within 2 days)

model is too simple and the clusters highly overlap. The user cannot analyze such huge result sets. In our approach we permit an overlap of clusters in a more meaningful manner by determining only the most interesting and non-redundant clusters. A detailed analysis of the Arxiv data reveals the basic problem: While in our approach just 4 vertices are included in more than 10 clusters, in CoPaM 781 vertices are in more than 10 clusters. CoPaM generates highly overlapping clusters; thus, generating redundant information. Our approach, however, realizes an overlap of clusters in a more meaningful manner.

Furthermore, Fig. 16 indicates that our approach generates clusters with just slightly less objects than CoPaM that concentrates on maximizing the number of vertices per cluster. The difference between the average cluster sizes of CoPaM and GAMer is continuously smaller than one standard deviation. Although GAMer implements a trade-off between different criteria it determines clusters of comparable size and at the same time achieves considerably higher density. The average density of the clusters detected by CoPaM is significantly smaller than the ones of GAMer even when taking the standard deviation into account. Additionally, for the gene data, we see that the dimensionality of GAMer's clusters is higher, too.

In Fig. 17, we analyze the trade-off in our model by varying the quality parameters a and b. In each case we vary just one parameter (a or b respectively) while fixing the remaining parameters to 1. Figure 17 indicates that increasing the parameter a yields an increase of the average density of the clustering result (solid circles, left axis). Consequently, varying parameter b respectively, results in an increase of the average number of vertices per cluster (non-solid circles, right axis). Our quality function enables us to control the clustering result based on the users' notion of interest. Our model offers a high flexibility.

5.5 Effectivity of pruning the set enumeration tree

In this section, we analyze how much the different pruning techniques contribute to the efficiency of GAMer. For this reason, we run GAMer several times on the same dataset with the same parameters but each time ignoring one of the pruning techniques. In Fig. 18 we compare the runtime of the different runs. The techniques for pruning by vertex degree, upper bound and lower bound are summarized here by the term 'vertex-based pruning techniques'.

For this experiment, we use a synthetic dataset which consists of 996 vertices with 3542 edges, containing 80 clusters and has an overall dimensionality of 20. The cluster sizes vary between 10 and 15 vertices, the number of relevant dimensions between 2 and 5 and the

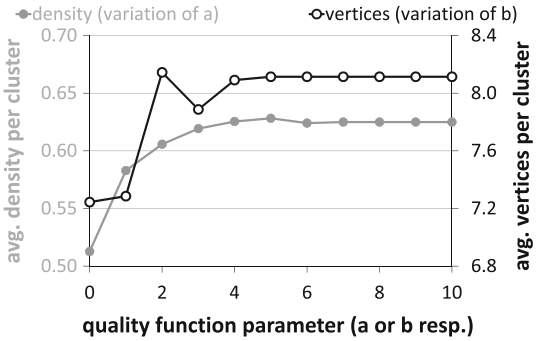


Fig. 17 Influence of the quality function on the clustering result (Arxiv data)

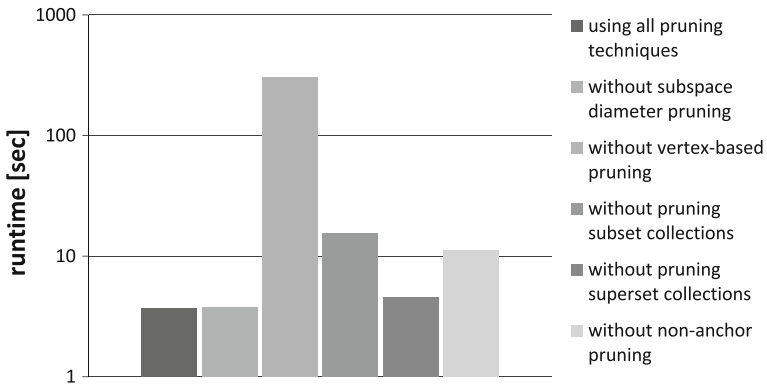


Fig. 18 Influence of the different pruning techniques

densities between 0.4 and 0.5. The quality parameters for all runs are set to $a = c = 0$ and $b = 1$, thus clusters with a large number of vertices are preferred over smaller ones.

Efficiency gain of the different pruning techniques As shown in Fig. 18, using all pruning techniques simultaneously (first bar) leads to the lowest runtime. The effect of the pruning techniques that are based on the cluster definition is analyzed in the following: By leaving out the pruning by subspace diameter (second bar) the runtime increases only by a small amount (ca. 2 %). Because the vertex-based pruning techniques often exclude most of the vertices that are also excluded by the subspace diameter pruning, we only get slightly bigger candidate sets by skipping the subspace diameter pruning. In the next run (bar 3) we do not use the vertex-based pruning techniques. The runtime is now two orders of magnitude higher than the runtime for using all pruning techniques. This experiment shows that the vertex-based pruning techniques highly contribute to the efficiency of GAMer.

Next, we analyze the effect of the pruning techniques that are based on the clustering definition: In this setting, the pruning of subset collections (bar 4) provides a large efficiency gain. Because of the parameters $a = c = 0$ and $b = 1$, large clusters get a higher quality than their subsets, even if the subsets have a higher density or dimensionality. As the analyzed technique aims at excluding redundant subsets it is especially useful in such parameter settings. Contrarily, the pruning of superset collections (bar 5) only provides a small efficiency gain here, as in our parameter setting a superset will never be redundant with respect to one of its subsets. The last bar shows the effect of the pruning with respect to non-anchor clusters. As shown in Fig. 18, this pruning technique also leads to a considerable efficiency gain.

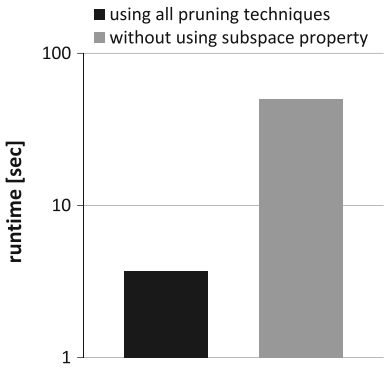


Fig. 19 Pruning using the subspace cluster property

Efficiency gain of pruning by subspace property Last we analyze the effect of jointly using the subspace cluster property and the quasi-clique property for pruning. In Sects. 4.1.3 and 4.1.4, we developed several pruning techniques incorporating the attribute similarity into graph based pruning. In Fig. 19, we compare the runtime of GAMer using this specialized pruning based on the subspace property with the runtime by neglecting this property. For the latter case, GAMer simply applies the vertex-based pruning techniques with the traditional indeg/exdeg definitions which consider just the graph information. Furthermore, also the pruning by subspace diameter uses the traditional definition for the k-neighborhood which does not take the attributes into account. The figure shows that by ignoring the attribute values the runtime increases by an order of magnitude. Thus, the combined usage of graph and attribute information is crucial for an efficient mining of twofold clusters.

6 Conclusion

We introduced the method GAMer for finding homogeneous groups of objects regarding combined graph and attribute data. Our twofold clusters join the advantages of subspace clustering and dense subgraph mining. We simultaneously account for the density, the size and the number of relevant dimensions for each cluster to obtain the most interesting ones. Our redundancy model confines the clustering by excluding redundant clusters that provide no additional information. Overall, we include only the most interesting and non-redundant clusters. Our GAMer exploits several novel pruning strategies based on the cluster definition and the redundancy model for efficiently determining this clustering result. Thorough experiments demonstrate that GAMer constantly outperforms the competing approaches in terms of efficiency and clustering quality. For repeatability and further research, we provide all datasets, measures, executables, and parameter settings on our website.

References

1. Abello J, Resende M, Sudarsky S et al. (2002) Massive quasi-clique detection. Lecture Notes in Computer Science pp. 598–612
2. Aggarwal C, Wang H (2010) Managing and mining graph data. Springer, New York
3. Aggarwal C, Wolf J, Yu P, Procopiuc C, Park J (1999) Fast algorithms for projected clustering. In: SIGMOD, pp 61–72
4. Al Hasan M, Chaoji V, Salem S, Besson J, Zaki M (2007) Origami: mining representative orthogonal graph patterns. In: ICDM, pp 153–162

5. Beyer KS, Goldstein J, Ramakrishnan R, Shaft U (1999) When is "nearest neighbor" meaningful?. In: ICDT, pp 217–235
6. Condon A, Karp RM (2001) Algorithms for graph partitioning on the planted partition model. Random Struct Algorithms 18(2):116–140
7. Ding CHQ, He X, Zha H, Gu M, Simon HD (2001) A min-max cut algorithm for graph partitioning and data clustering. In: ICDM, pp 107–114
8. Du N, Wu B, Pei X, Wang B, Xu L (2007) Community detection in large-scale social networks. In: WebKDD/SNA-KDD, pp 16–25
9. Ester M, Ge R, Gao BJ, Hu Z, Ben-Moshe B (2006) Joint cluster analysis of attribute data and relationship data: the connected k-center problem. In: SDM
10. Garey M, Johnson D (1979) Computers and intractability: a guide to NP-completeness. W.H Freeman and Company, San Francisco
11. Günnemann S, Färber I, Boden B, Seidl T (2010) Subspace clustering meets dense subgraph mining: a synthesis of two paradigms. In: ICDM, pp 845–850
12. Günnemann S, Färber I, Müller E, Assent I, Seidl T (2011) External evaluation measures for subspace clustering. In: CIKM, pp 1363–1372
13. Günnemann S, Kremer H, Seidl T (2010) Subspace clustering for uncertain data. In: SDM, pp 385–396
14. Günnemann S, Müller E, Färber I, Seidl T (2009) Detection of orthogonal concepts in subspaces of high dimensional data. In: CIKM, pp 1317–1326
15. Han J, Kamber M (2006) Data mining: concepts and techniques. Morgan Kaufmann, San Francisco
16. Hanisch D, Zien A, Zimmer R, Lengauer T (2002) Co-clustering of biological networks and gene expression data. Bioinformatics 18:145–154
17. Jolliffe I (2002) Principal component analysis, 2nd edn. Springer, New York
18. Kailing K, Kriegel HP, Kroeger P (2004) Density-connected subspace clustering for high-dimensional data. In: SDM, pp 246–257
19. Kriegel HP, Kröger P, Zimek A (2009) Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. TKDD 3(1):1–58
20. Kubica J, Moore AW, Schneider JG (2003) Tractable group detection on large link data sets. In: ICDM, pp 573–576
21. Liu G, Wong L (2008) Effective pruning techniques for mining quasi-cliques. In: ECML/PKDD (2). pp 33–49
22. Long B, Wu X, Zhang ZM, Yu PS (2006) Unsupervised learning on k-partite graphs. In: KDD, pp 317–326
23. Long B, Zhang ZM, Yu PS (2007) A probabilistic framework for relational clustering. In: KDD, pp 470–479
24. Moise G, Sander J (2008) Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In: KDD, pp 533–541
25. Moise G, Sander J, Ester M (2006) P3C: a robust projected clustering algorithm. In: ICDM, pp 414–425
26. Moser F, Colak R, Rafiey A, Ester M (2009) Mining cohesive patterns from graphs with feature vectors. In: SDM, pp 593–604
27. Müller E, Assent I, Günnemann S, Krieger R, Seidl T (2009) Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In: ICDM, pp 377–386
28. Müller E, Günnemann S, Assent I, Seidl T (2009) Evaluating clustering in subspace projections of high dimensional data. In: VLDB, pp 1270–1281
29. Neville J, Adler M, Jensen D (2004) Spectral clustering with links and attributes. Dept of Computer Science, University of Massachusetts Amherst, Tech. rep
30. Parsons L, Haque E, Liu H (2004) Subspace clustering for high dimensional data: a review. SIGKDD Explor 6(1):90–105
31. Pei J, Jiang D, Zhang A (2005) On mining cross-graph quasi-cliques. In: KDD, pp 228–238
32. Procopiuc CM, Jones M, Agarwal PK, Murali TM (2002) A monte carlo algorithm for fast projective clustering. In: SIGMOD, pp 418–427
33. Ruan J, Zhang W (2007) An efficient spectral algorithm for network community discovery and its applications to biological and social networks. In: ICDM, pp 643–648
34. Rymon R (1992) Search through systematic set enumeration. In: K.R., pp 539–550
35. Sequeira K, Zaki MJ (2004) Schism: a new approach for interesting subspace mining. In: ICDM, pp 186–193
36. Shiga M, Takigawa I, Mamitsuka H (2007) A spectral clustering approach to optimally combining numerical vectors with a modular network. In: SIGKDD, pp 647–656
37. Shyamsundar R, et al. (2005) A DNA microarray survey of gene expression in normal human tissues. Genome Biol 6(3):R22

38. Silva A, Meira W Jr, Zaki M (2010) Structural correlation pattern mining for large graphs. In: Workshop on mining and learning with graphs, pp 119–126
39. Stark C, et al. (2006) BioGRID: a general repository for interaction datasets. Nucleic Acids Res 34(suppl 1):D535–D539
40. Ulitsky I, Shamir R (2007) Identification of functional modules using network topology and high-throughput data. BMC Syst Biol 1(1):8
41. Wang J, Zeng Z, Zhou L (2006) Clan: an algorithm for mining closed cliques from large dense graph databases. In: ICDE, p 73
42. Yiu ML, Mamoulis N (2003) Frequent-pattern based iterative projected clustering. In: ICDM, pp 689–692
43. Yiu ML, Mamoulis N (2005) Iterative projected clustering by subspace mining. IEEE Trans Knowl Data Eng (TKDE) 17(2):176–189
44. Zeeberg B, et al (2003) GoMiner: a resource for biological interpretation of genomic and proteomic data. Genome Biol 4(4):R28
45. Zeng Z, Wang J, Zhou L, Karypis G (2006) Coherent closed quasi-clique discovery from large dense graph databases. In: KDD, pp 797–802
46. Zeng Z, Wang J, Zhou L, Karypis G (2007) Out-of-core coherent closed quasi-clique mining from large dense graph databases. TODS 32(2):13
47. Zhang S, Yang J, Li S (2009) RING: an integrated method for frequent representative subgraph mining. In: ICDM, pp 1082–1087
48. Zhou Y, Cheng H, Yu JX (2009) Graph clustering based on structural/attribute similarities. In: VLDB, pp 718–729
49. Zhou Y, Cheng H, Yu JX (2010) Clustering large attributed graphs: an efficient incremental approach. In: ICDM, pp 689–698

Author Biographies

Stephan Günnemann is a Postdoctoral researcher at the Department of Computer Science, Carnegie Mellon University, USA. From 2008 to 2012, he has been a research associate at the data management and data exploration group at RWTH Aachen University, Germany. His research interests include graph mining and the mining of non-redundant, multiple clustering solutions for high-dimensional and structured data. Dr. Günnemann received his Diplom (MSc) in 2008 and his Ph.D. in 2012 from RWTH Aachen University.

Ines Färber is a Ph.D. student and research assistant in computer science at the RWTH Aachen University, Germany. Her research interests include detection of clusters in subspace projections of high-dimensional data, mining of alternative and multi-view clustering solutions, and clustering for graph data. Ines Färber received her Diplom (MSc) in 2009 from RWTH Aachen University.

Brigitte Boden is a Ph.D. student and research assistant in computer science at the RWTH Aachen University, Germany. Her research interests include data mining and knowledge discovery, especially on network data. Brigitte Boden received her Diplom (MSc) in 2010 from RWTH Aachen University.

Thomas Seidl is a professor for computer science and head of the data management and data exploration group at RWTH Aachen University, Germany. His research interests include data mining and database technology for multimedia and spatio-temporal databases in engineering, communication and life science applications. Prof. Seidl received his Diplom (MSc) in 1992 from TU Muenchen and his Ph.D. (1997) and venia legendi (2001) from LMU Muenchen.