Subspace Clustering for Complex Data

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Stephan Günnemann
aus Meppen

Berichter: Univ.-Prof. Dr. rer. nat. Thomas Seidl
Prof. Dr. sc. techn. Martin Ester

Tag der mündlichen Prüfung: 15.03.2012

Diese Dissertation ist auf den Internetseiten
der Hochschulbibliothek online verfügbar.

Contents

Abstract / Zusammenfassung **1**

1 Overview of Thesis **5**
1.1 Introduction 5
1.2 Subspace Clustering for Complex Data 8
1.3 Contributions and Structure of this Thesis 12

I Subspace Mining on Vector Data **17**

2 Relevant Subspace Clustering **19**
2.1 Motivation and Comparison with Related Work 20
2.2 Relevant Subspace Clustering 22
2.3 The RESCU Algorithm 36
2.4 Experimental Analysis 40
2.5 Conclusion 46

3 Detection of Orthogonal Concepts **47**
3.1 Motivation 48
3.2 Comparison with Related Work 50
3.3 Orthogonal Concepts in Subspaces 52
3.4 The OSCLU Algorithm 61
3.5 Experimental Analysis 66
3.6 Conclusion 71

4 Multi-View Subspace Clustering via Multiple Mixture Models **73**
4.1 Motivation and Comparison with Related Work 74
4.2 Formal Description of the Model 77
4.3 The MVGen Algorithm 84

4.4 Experimental Analysis 94
4.5 Conclusion 101

5 Subspace Correlation Clustering 103
5.1 Motivation 103
5.2 Comparison with Related Work 107
5.3 Subspace Correlation Clustering 109
5.4 The SSCC Algorithm 117
5.5 Experimental Analysis 122
5.6 Conclusion 129

II Subspace Mining on Imperfect Data 131

6 Imperfect Data: Introduction 133
6.1 Types of Imperfect Information 133
6.2 Data Cleansing 135

7 Subspace Clustering for Incomplete Data 137
7.1 Motivation and Comparison with Related Work 138
7.2 Fault Tolerant Subspace Clustering Model 140
7.3 Instantiation and FTSC Algorithm 151
7.4 Experimental Analysis 155
7.5 Conclusion 160

8 Subspace Clustering for Uncertain Data 161
8.1 Motivation 161
8.2 Comparison with Related Work 164
8.3 Subspace Clustering for Uncertain Data 165
8.4 Algorithmic Properties 177
8.5 Experimental Analysis 180
8.6 Conclusion 187

III Subspace Mining on Heterogeneous Data 189

9 Heterogeneous Data: Introduction 191
9.1 Data Heterogeneity 191

9.2 Mining Diversity 192

10 A Synthesis of Subspace Clustering and Dense Subgraph Mining 195
10.1 Motivation 196
10.2 Comparison with Related Work 199
10.3 An Integrated Clustering Model 201
10.4 The GAMER Algorithm 211
10.5 Experimental Analysis 232
10.6 Conclusion 241

11 Efficient Mining of Combined Subspace and Subgraph Clusters 243
11.1 Motivation 243
11.2 Comparison with Related Work 246
11.3 Quality-Optimal Clustering 246
11.4 The EDCAR Algorithm 255
11.5 Experimental Analysis 263
11.6 Conclusion 268

12 Density-Based Subspace Clustering for Graph Data 269
12.1 Motivation and Comparison with Related Work 269
12.2 A Density-Based Clustering Model for Combined Data 272
12.3 The DB-CSC Algorithm 282
12.4 Experimental Analysis 293
12.5 Conclusion 296

IV Summary 299

13 Conclusion and Future Work 301
13.1 Conclusion 301
13.2 Future Work 304

Appendices I

Bibliography III
List of Publications XXIII
Curriculum Vitae XXIX

Abstract

The increasing potential of storage technologies and information systems has opened the possibility to conveniently and affordably gather large amounts of complex data. Going beyond simple descriptions of objects by some few characteristics, such data sources range from high dimensional vector spaces over imperfect data containing errors to network data describing relations between the objects. Data Mining is the task of extracting previously unknown and useful patterns from such data sources by using automatic or semi-automatic algorithms. In this thesis, we focus on the mining task of *clustering*, which aims at grouping similar objects while separating dissimilar ones.

Since in today's applications usually many characteristics for each object are recorded, one cannot expect to find similar objects by considering all attributes together. In contrast, valuable clusters are hidden in subspace projections of the data. As a general solution to this problem, the paradigm of *subspace clustering* has been introduced, which aims at automatically determining for each group of objects a set of relevant attributes these objects are similar in.

In this thesis, we introduce novel methods for effective subspace clustering on various types of complex data. Our methods tackle major open challenges for clustering in subspace projections. We study the problem of redundancy in subspace clustering results and propose models whose solutions contain only non-redundant and, thus, valuable clusters. Since different subspace projections represent different views on the data, often several groupings of the objects are reasonable. Thus, we propose techniques that are not restricted to a single partitioning of the objects but that enable the detection of multiple clustering solutions. Besides tackling these challenges of subspace clustering for the case of vector data, we study the task of subspace clustering on two further data types: imperfect data and network data in combination with vector data. We propose integrated mining techniques directly handling errors in the data and simultaneously mining different information sources. In thorough experiments, we demonstrate the strengths of our novel clustering approaches. Overall, for the first time, meaningful subspace clustering results can be obtained for these types of complex data.

Zusammenfassung

Das gestiegene Potential von Speichertechnologien und Informationssystemen hat die Möglichkeit eröffnet, kostengünstig große Mengen an komplexen Daten zu sammeln. Neben einfachen Beschreibungen von Objekten durch einige wenige Attribute reichen diese Datenquellen von hochdimensionalen Vektorräumen über unvollkommene Daten hin zu Netzwerkdaten. Die Aufgabe des Data Mining ist es, mit Hilfe von automatischen oder semi-automatischen Algorithmen aus diesen Datenquellen bislang unbekannte und nützliche Muster zu extrahieren. In dieser Arbeit betrachten wir die Aufgabe des *Clusterings*, die darauf abzielt Gruppen von ähnlichen Objekten zu bilden und gleichzeitig unähnliche Objekte zu trennen.

Da in heutigen Anwendungen häufig sehr viele Eigenschaften für jedes Objekt gespeichert werden, ist nicht zu erwarten, dass Objekte existieren, die bei Betrachtung der Gesamtheit aller Eigenschaften ähnlich zueinander sind. Vielmehr werden sinnvolle Gruppen nur in Teilräumen des Datenraums gefunden. Als Lösung für dieses Problem wurde das Paradigma des *Subspace Clusterings* eingeführt, welches automatisch für jede Gruppe von Objekten eine zugehörige Menge relevanter Attribute identifiziert, in welchen die Objekte ähnlich zueinander sind.

In dieser Arbeit führen wir neue Methoden für ein effektives Subspace Clustering auf verschiedenen Typen von komplexen Daten ein. Wir untersuchen das Problem der Redundanz in Subspace Clustering-Ergebnissen und schlagen neue Modelle zur Vermeidung dieser Redundanz vor. Da jeder Teilraum eine andere Sicht auf die Daten liefert, können häufig mehrere sinnvolle Gruppierungen der Objekte gefunden werden. Daher führen wir Techniken ein, die nicht auf eine einzige Partitionierung der Objekte eingeschränkt sind sondern mehrere unterschiedliche Gruppierungen finden können. Neben der Lösung dieser Herausforderungen für das Subspace Clustering von vektoriell beschriebenen Daten analysieren wir ferner das Subspace Clustering auf unvollkommenen Daten sowie auf einer Kombination von Netzwerkdaten mit vektoriellen Daten. Wir schlagen integrierte Analysetechniken vor, welche mit Fehlern in den Daten umgehen können und verschiedene Datenquellen simultan analysieren. In experimentellen Untersuchungen zeigen wir die Stärken der neu entwickelten Clustering-Methoden. Insgesamt ermöglichen wir erstmalig die Bestimmung eines sinnvollen Subspace Clustering für diese komplexen Daten.

Chapter 1

Overview of Thesis

1.1 Introduction

The increasing potential of storage technologies and information systems over the last decades has opened the possibility to conveniently and affordably gather large amounts of complex data. Going beyond simple descriptions of objects by some few characteristics, such data sources range from high dimensional vector spaces over imperfect data containing errors to network data describing relations between the objects. While storing these data is common, their analysis is challenging: the human capabilities of a manual analysis are quickly exhausted considering the mere size of the data. Thus, automatic techniques supporting the user in the process of knowledge extraction are required to gain a benefit from the collected data.

The concept of Knowledge Discovery in Databases (KDD) [HK01] has been evolved as a possible solution for the above challenge and it is coherently described by a multilevel process the user has to follow (cf. Figure 1.1). Given the raw data, which is rarely perfect since, e.g., missing entries, inconsistencies, or uncertain values are prevalent during the data acquisition phase, the KDD process starts with a preprocessing step to clean the data. This step is often referred to data cleansing and tries to increase the data quality to support the subsequent data mining step. The goal of data mining, as the key component of the KDD process, is to extract previously unknown and useful patterns from the data using automatic or semi-automatic algorithms. Finally, the KDD process concludes with the pre-

Figure 1.1: Knowledge Discovery in Databases (KDD) process

sentation and evaluation of the detected patterns, enabling the user to understand and interpret the results.

In this thesis we focus on the development of novel models and algorithms for the central step of the KDD process: data mining. Out of the several mining tasks that exist in the literature, this work centers on the important method of *clustering*, which aims at grouping similar objects while separating dissimilar ones. Clustering, as an unsupervised learning task, analyses data without given labels but automatically reveals the hidden structure of the data by its aggregations. For today's data, however, it is known that traditional clustering methods fail to detect meaningful patterns. The problem originates from the fact that traditional clustering approaches consider the full space to measure the similarity between objects, i.e. all characteristics of the objects are taken into account. While collecting more and more characteristics, however, it is very unlikely that two objects are similar with respect to the full space and often some dimensions are not relevant for clustering. A continuative aspect is the decreasing discrimination power of distance functions with increasing dimensionality of the data space due to the "curse of dimensionality" [BGRS99]. 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.

Global dimensionality reduction techniques, e.g., based on Principal Component Analysis (PCA [Jol02]), try to mitigate these effects, but they do not provide a solution to this problem. Since they reduce all objects to a single projection, they cannot detect clusters with locally relevant dimensions. In complex data sets, however, different groups of objects may have different relevant dimensions. In Figure 1.2, the objects depicted as rectangles are similar in a 2-dimensional sub-

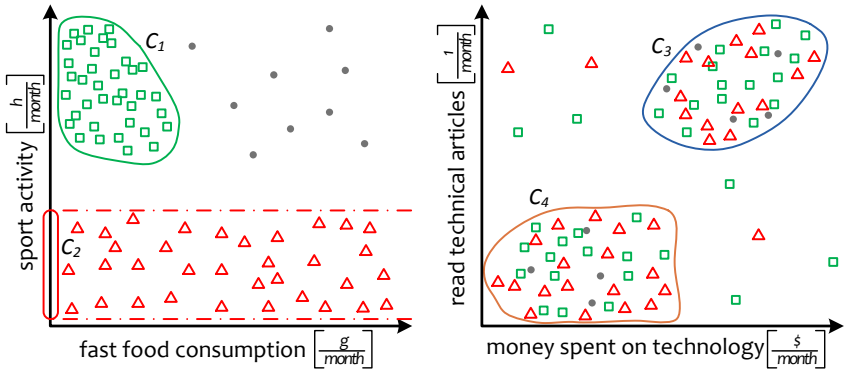


Figure 1.2: Exemplary subspace clustering of a 4-dimensional database

space, while the objects depicted as triangles show only similar values in a single dimension.

As a general solution to this problem, the paradigm of *subspace clustering* [PHL04, KKZ09] has been introduced. Subspace clustering detects clusters in arbitrary subspace projections of the data by automatically determining for each group of objects a set of relevant dimensions these objects are similar in. Thus, in Figure 1.2 the objects grouped in cluster C_1 would correspond to a subspace cluster in subspace {fast food consumption, sport activity}, while the cluster C_2 is only located in subspace {sport activity}. Since different subspaces may lead to different groupings, each object can naturally belong to multiple clusters as illustrated in Figure 1.2 (right). The subspaces individually assigned to each group provide the reasoning why such multiple solutions are meaningful. Thus, in the example of Figure 1.2, each of the four clusters $\{C_1, \ldots, C_4\}$ is useful and should be provided to the user.

In this thesis we introduce novel methods for effective subspace clustering on complex data including high-dimensional vector spaces, imperfect data, and heterogeneous data. Such clustering methods are beneficial for various applications: In customer and social network analysis, persons can be grouped according to their similarity based on some product relevant attributes. In bioinformatics, groups of genes that show similar expression levels in a subset of experimental medical treatments can be identified. In sensor network analysis, different environmental events can be described by similarly behaving sensors with respect to specific measured variables. For all of these domains objects are characterized by many attributes, while the clusters appear only in subspace projections of the data.

1.2 Subspace Clustering for Complex Data

In the previous section we introduced the basic objective of subspace clustering. Though, achieving this objective, especially for today's complex data, is subject to several challenges, which we will discuss in this section. These challenges affect different levels of the KDD process and in total they support the demand for novel, effective subspace clustering models as developed in this thesis.

Challenges for Subspace Clustering on Vector Data

Redundancy. As the example in Figure 1.2 indicates, subspace clustering finds groups of objects in arbitrary projections of the data. A naive approach for subspace clustering would be to apply traditional clustering on any possible subspace projection. However, besides the high computational demand due to the exponential number of subspaces w.r.t. the number of dimensions that have to be analyzed, this approach generates results with a tremendous amount of redundant clusters. In Figure 1.2 for example, one would detect that the objects of the 2-dimensional subspace cluster C_1 are also similar in the 1-dimensional projections {sport activity} and {fast food consumption}; this results in already three clusters. Though, most of these groups do not provide novel knowledge about the data's structure since, e.g., the clustered objects among these groups are identical. A challenge for subspace clustering is to avoid such redundant information in the clustering result and thus presenting the user a result of manageable size.

Multiple views. Eliminating redundancy from subspace clustering results has to be regarded carefully: overlapping clusters are not necessarily a sufficient criterion for redundancy. Since different subspaces represent different views on the data, objects are allowed to be contained in several clusters without inducing redundancy (cf. Figure 1.2). The subspace clusters of each view provide novel information about the data's characteristic, and their grouping into views enables further interpretations about the clusters' interrelations. For effective subspace clustering, we have to tackle the challenge of detecting the data's multiple views and the underlying subspace clusters.

Complex patterns. In general, clustering aims at grouping similar objects. While many techniques constrain themselves on detecting dense areas, i.e. similarity of objects is reflected in similar attribute values, one also observes more complex patterns. A prominent example for such a pattern is the correlation be-

	d1	d2	d3	d4	d5
o1	low value	low value	low value		
o2	low value	low value	low value		
o3	low value	low value	low value		
o4	low value	low value	???	high value	high value
o5			high value	high value	high value
o6			high value	high value	high value
o7			high value	high value	high value

Figure 1.3: Challenge of handling missing values in subspace clustering

tween attributes, i.e. a set of objects is regarded as similar if their attribute values depend on each other in a similar way. Detecting such complex patterns in subspace projections is a challenge effective subspace clustering methods have to cope with.

Overall, the previous challenges deal with level 3 of the KDD process by focusing on the set of detected patterns. Though, the basic requirement such enhancements rely on is a cleaned data set as provided in level 2 of the KDD process. The challenges accompanied by this step are discussed in the next section.

Challenges for Subspace Clustering on Imperfect Data

Most algorithms assume perfect data as input. Imperfect information, however, is ubiquitous where data is recorded: Missing values, for example, occur due to sensor faults in sensor networks, or uncertainty about attribute values is present due to noise or privacy issues. There is a need to handle such imperfect data for the task of subspace clustering.

Integrated mining. Naively, traditional data cleansing techniques could be applied to preprocess the data before clustering. This procedure, however, has a major drawback: preprocessing methods are not aware of the specific characteristics introduced for subspace clustering as, e.g., the occurrence of objects in multiple clusters. Considering the exemplary database illustrated in Figure 1.3, the object o_4 should be included in both subspace clusters since dependent on the currently considered subspace we are able to find an instantiation of the missing value such that o_4 is similar to the remaining objects of the cluster. Traditional preprocessing, however, is not aware of different subspaces; it would instantiate the missing entry by a single value, and at least one cluster would not be detected

Figure 1.4: Enhanced KDD process by integrating the preprocessing step into the mining step for better handling of imperfect data

correctly. Thus, preprocessing leads to an information loss if used in combination with subspace clustering.

For an effective subspace clustering on imperfect data, we have to perform an integrated mining. By joining the preprocessing step with the actual mining task, we are able to account for the special characteristics of subspace clustering leading to a better handling of imperfect information. Figure 1.4 illustrates this integrated mining step: instead of mining the preprocessed data, the mining method directly analyzes the raw data and, e.g., instantiates missing values based on the currently detected subspace clusters.

Robust clustering models. Directly operating on imperfect data leads to novel requirements for subspace clustering models and definitions ranging from the accurate determination of similarity values between individual objects to the overall coherence of a cluster in an imperfect setting. The underlying challenge to be tackled by these models is their robustness against 'errors' in the data. Even for a high-degree of imperfect information, reliably detecting high quality patterns should be possible.

In summary, the issues discussed so far focus on level 2 and 3 of the KDD process, and they assume a single representation of the objects by, e.g., a database of high-dimensional vectors. As the following section illustrates, the assumption of single data type has to be relaxed in many applications; we also have to adapt level 1 of the KDD process.

Challenges for Subspace Clustering on Heterogeneous Data

Traditional data mining algorithms process just a single type of data, e.g., objects embedded into a vector space. Today's applications, however, can acquire multiple, diverse, and heterogeneous data sources. Besides characterizing single objects

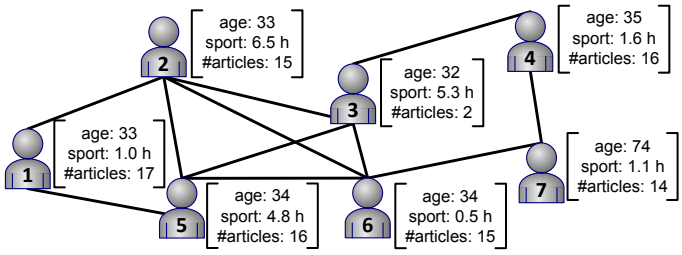


Figure 1.5: Exemplary social network represented by vector and graph data

by vector data, network information, for example, is a ubiquitous source to indicate the relations between different objects. Such type of heterogeneous data can be observed in various domains including social networks, where friendship relationships are available along with the users' individual interests (cf. Figure 1.5); systems biology, where interacting genes and their specific expression levels are recorded; and sensor networks, where connections between the sensors as well as individual measurements are given. While imperfect data, as introduced in the previous section, usually reduces the amount of information that can be utilized in the mining step, heterogeneous data increases this pool. Thus, to realize the full potential for knowledge extraction, mining techniques should consider all available information sources. In this thesis we focus on heterogeneous data as given in the applications mentioned above because this type is one of the most prevalent ones. The general challenges for subspace clustering, however, apply for all types of heterogeneous data.

Integrated mining. A sequential process for heterogeneous data, which first mines each type independently and then compares the detected patterns, is problematic since the results of each source might differ or even contradict. Thus, for an effective subspace clustering, again an integrated mining promises more meaningful and accurate results. By simultaneously mining different types of information, as illustrated in the adapted KDD process of Figure 1.6, inaccurate information in one source can be mitigated by the other sources and an overall coherent result is possible.

By integrating heterogeneous information types into a single mining task, different paradigms as graph mining and subspace clustering need to be joined. Since each paradigm has established its own objectives and even worse these objectives might contradict, e.g., large clusters vs. high dimensional clusters, a sound synthesis is highly challenging but actually crucial. We have to ensure that the

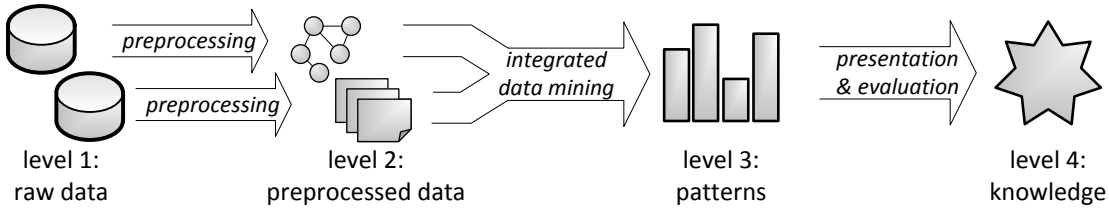


Figure 1.6: Enhanced KDD process by simultaneously mining multiple information types for better handling of heterogeneous data

paradigms are treated on an equal footing and none of the data sources is favored over the other. Especially retaining the characteristic of subspace clustering — objects might be part of multiple patterns potentially inducing redundancy — is a challenge posed by mining heterogeneous data.

Efficient algorithms. Since the databases to be analyzed and the resulting patterns to be detected are inherently complex due to the heterogeneous types of information, the algorithms' efficiency is an important aspect we have to cope with. Thus, developing novel pruning techniques or algorithms that efficiently provide an approximate mining result is a further requirement for today's subspace clustering methods.

Summary

As shown in this section, for an effective subspace clustering on complex data various challenges have to be solved and novel techniques need to be introduced covering the first three levels of the KDD process. Only such novel models ensure to detect meaningful patterns and, thus, potentially lead to new and useful knowledge.

1.3 Contributions and Structure of this Thesis

This thesis introduces models and algorithms for effective subspace clustering on complex data. To provide a holistic view on the KDD process, all aforementioned challenges are tackled in this work. The following section presents an overview of the major contributions and the general structure of this thesis while details are discussed in the subsequent chapters. The thesis is divided into four parts.

Part 1: Subspace Mining on Vector Data

In the first part of this thesis we focus on subspace clustering methods for vector data. As our first contribution, we introduce our clustering model RESCU that avoids redundant information in the final clustering in Chapter 2. In contrast to existing approaches that simply exclude lower dimensional projections of clusters, our method introduces a global optimization approach taking all clusters simultaneously into account. Our optimization ensures to select the most interesting clusters, where each cluster provides novel knowledge w.r.t. all other clusters in the final result. Unlike to projected clustering methods, which avoid redundancy by enforcing disjoint clusters, RESCU allows overlapping clusters in general by extending the Set Cover optimization problem to the subspace clustering paradigm. Our model is highly flexible since it is independent of the underlying subspace cluster definition and the selected interestingness measures. We prove that the computation of our RESCU model is NP-hard, and we propose an algorithm determining an approximate solution showing high clustering accuracy.

While RESCU successfully avoids redundant information, it is not able to detect multiple views in the data. Thus, as our second contribution, we introduce clustering models able to solve this challenge. In Chapter 3, we present our OSCLU method that judges the redundancy of clusters based on their similarity regarding objects and subspaces. The global optimization method of OSCLU actively includes novel knowledge of (almost) orthogonal subspaces into the final clustering result. While our OSCLU model provides a general and flexible solution to detect subspace clusters hidden in multiple views, we prove its complexity to be NP-hard and propose an efficient algorithm to compute an approximate solution.

OSCLU is able to detect clusters hidden in multiple views. The views itself, however, are not explicitly mined, i.e., it remains unknown which clusters belong to the same view and which attributes are characteristic for this view. To overcome these limitations, we introduce our MVGen method in Chapter 4. Based on a generative model that considers the data as a result of a process generated by different views, we couple the detection of subspace clusters and their aggregating views. Using our novel representation, we perform Bayesian model selection to determine those dimensions which are relevant for the views and their subspace clusters, and we enable the detection overlapping clusters and overlapping views without inducing redundancy. Since exact inference in our model is intractable, we

use the principle of iterated conditional modes to realize an efficient and effective clustering algorithm.

While the previous methods focus on clusters corresponding to dense areas in the data space, we propose in Chapter 5 a method which is able to detect more complex patterns. As our next contribution, we introduce the novel paradigm of subspace correlation clustering: we analyze *subspace projections* to find *subsets of objects* showing *linear correlations* among this subset of dimensions. While existing correlation clustering methods are limited to almost disjoint clusters, our model allows each object to contribute to several correlations due to different subspace projections. In our paradigm, we permit multiple overlapping clusters but simultaneously avoid redundant clusters deducible from already known correlations originating from collinearity or induction. Furthermore, as known from traditional clustering, analyzing the full-space might be questionable due to the decreasing discrimination power of distances resulting from multiple overlapping patterns and irrelevant dimensions. Therefore, by analyzing individual subspace projections, our model opens the potential to detect correlations that clearly stand out in the data. We develop the algorithm SSCC, which exploits different pruning techniques to efficiently generate a subspace correlation clustering result.

Part 2: Subspace Mining on Imperfect data

In the second part of this thesis we focus on subspace clustering methods for imperfect data. After introducing and classifying the various types of imperfect data in Chapter 6, we analyze two scenarios in detail: imperfect data due to missing values and imperfect data due to uncertainty about attribute values.

As our contribution in Chapter 7, we introduce a general fault tolerance definition enhancing subspace clustering models to handle missing values. Our model handles missing values based on the currently considered subspace and set of objects. It, thus, enables objects to be part of multiple subspace clusters. Our variable thresholds realize a flexible model that adapts to the individual cluster characteristics and ensures a robust parameterization. The high complexity of an integrated subspace clustering method handling missing values demands for effective pruning methods. Thus, we prove important monotonicity properties of our general model. In the experimental evaluation, our approach yields high quality results even in the presence of many missing values.

In Chapter 8 we present an integrated mining method for subspace clustering on uncertain data. We present three variants that can be distinguished by the amount of information extracted from the probability density functions. Since for subspace clustering and especially in uncertain scenarios a strict assignment of objects to single clusters is not appropriate, we enrich our model with the concept of membership degree enabling a non-partitioning clustering result. To cope with the computational challenges of subspace clustering for uncertain data, we propose an efficient solution that uses Apriori-based pruning and heuristic sampling. Thorough experimental evaluation demonstrates that our integrated subspace clustering method substantially outperforms competing techniques.

Part 3: Subspace Mining on Heterogeneous Data

In the third part of this thesis we develop subspace clustering methods for heterogeneous data. We discuss the diversity of data types and mining challenges in Chapter 9, before we focus on the development of integrated clustering techniques simultaneously handling network data in combination with vector data in the subsequent chapters.

In Chapter 10 we introduce our GAMER method for finding homogeneous groups in such heterogeneous data by joining the paradigms of subspace clustering and dense subgraph mining. GAMER detects clusters that are optimized according to their density, size, and number of relevant dimensions to fairly tradeoff the different paradigms' objectives and to obtain the most interesting clusters. GAMER confines the clustering by excluding redundant clusters but still allows multiple views on the data. By incorporating both data sources we develop novel pruning strategies that lead to an efficient calculation of our clustering.

Extending the GAMER method, we develop our EDCAR model in Chapter 11. EDCAR performs a global optimization to determine the overall clustering solution and to avoid redundancy. As shown for traditional vector data by our RESCU model, optimizing the clustering globally is beneficial but is usually more complex. We prove the complexity of our EDCAR model and identify the critical parts inhibiting an efficient execution. Based on this analysis, we develop an efficient and effective algorithm that approximates the optimal clustering solution. By interweaving the process of cluster generation and cluster selection, which both make use of the GRASP principle, we determine high quality clusters and ensure low runtimes.

As our last contribution in this part, we introduce in Chapter 12 the DB-CSC model that adopts a density-based cluster definition to take the attribute similarity in subspaces and the graph density into account. Even though the previously proposed approaches successfully overcome the problem of full-space clustering, their limited cluster definitions are restricted to clusters of certain shapes. Thus, introducing the novel notion of local densities, our DB-CSC model aims at detecting clusters of arbitrary shape and size. In contrast to density-based methods for vector data, calculating clusters in heterogeneous data is more challenging and we show how a fixed point iteration can be used to find the clusters. Using this principle and further pruning techniques, our DB-CSC algorithm efficiently determines the combined clustering solution.

Part 4: Summary

In the last part, we conclude this thesis by summing up all contributions and by presenting open challenges for future work in the area of subspace clustering.

Part I

Subspace Mining on Vector Data

Chapter 2

Relevant Subspace Clustering

In high-dimensional vector spaces, clusters rarely show up in the full dimensional space but are hidden in subspace projections of the data. Subspace clustering methods try to detect these patterns by analyzing arbitrary subspaces of the data for their clustering structure. In general, a subspace cluster $C = (O, S)$ is defined by a set of objects $O \subseteq DB$ that are similar in a subset of dimensions $S \subseteq Dim$.

As discussed in Chapter 1.2, naively performing subspace clustering yields to an overwhelming result of redundant clusters: often the objects grouped in cluster $C = (O, S)$ are also similar in the subspace projections $S' \subseteq S$. Since the number of possible subspace projections is exponential in the number of dimensions, traditional subspace clustering approaches generate a tremendously large result and they fail to detect only the relevant subspace clusters.

In this chapter, we propose our novel model for relevant subspace clustering (RESCU), detecting the most interesting non-redundant subspace clusters. In contrast to existing methods, our global redundancy elimination checks the relevance of each cluster against all other possible subspace clusters. It aims at reporting as few clusters as possible to reduce redundancy, yet include as many interesting clusters as possible. The general idea is to cover almost all objects (except of noise) by interesting clusters. Thus, we allow overlapping subspace clusters in general but require each cluster to contribute at least some novel knowledge by grouping additional objects. Overall, the result is optimized based on the user's specified interest and our global redundancy notion.

2.1 Motivation and Comparison with Related Work

In the last decades several methods for clustering in subspace projections were proposed. In general, we can distinguish two different paradigms: subspace clustering and projected clustering.

Subspace clustering. Subspace clustering detects clusters in arbitrary projections by automatically determining a set of relevant dimensions for each cluster [PHL04, KKZ09]. Thus, one is able to detect objects as part of various clusters in different subspaces. In bioinformatics, for example, genome data analysis clusters genes (objects) that show similar expression levels in a subset of experimental medical treatments (attributes). Such similarities might indicate functional relationships. Each gene might appear in multiple roles (subspace clusters). As a consequence, subspace clusters might overlap in the sense that they share objects. Starting with the first subspace clustering approach CLIQUE [AGGR98], recent research has seen a number of approaches using different definitions of what constitutes a subspace cluster [AGGR98, KKK04, SZ04, KKRW05, NGC01]. As summarized in a recent evaluation study [MGAS09], their common problem is that the output generated is typically huge.

Subspace clustering allows clusters to overlap, but has to cope with the detection of exponentially many subspace clusters in arbitrary projections. Many of these clusters do not provide any further information, as more or less the same object groups are detected in multiple projections of the data. Such redundant subspace clusters should be removed and only the most interesting ones, which provide novel knowledge about the data should be reported.

Some approaches towards modeling and removing redundant subspace clusters have been proposed. They define non-redundant and possibly overlapping subspace clusters, but only with a local scope. In [AKMS07, AKMS08a, AKMS08b], a subspace cluster is redundant if it shares a certain fraction of objects with another one. Retaining only maximal subspace clusters, i.e. the highest dimensional one of two, results in a clear increase in clustering quality. This is due to the fact that maximal clusters tend to contain less noise and thus represent the inherent data structure more faithfully. This definition of redundancy, however, is limited in two respects. First, redundancy is based only on a pairwise comparison of clusters. And second, the redundancy check incorporates only the fraction of jointly detected objects [AKMS07, AKMS08a, AKMS08b]. We call this a local redundancy definition, as only local properties like object count and comparison of two sub-

space clusters is used. Consequently, a redundant subspace cluster that is covered by a *combination* of high dimensional subspace clusters is still reported as non-redundant. In contrast, our model aims at a global redundancy check including a more flexible interestingness definition comparing each cluster with the overall set of detected subspace clusters.

Another subspace clustering approach aims at extracting non-redundant axis-parallel subspace regions [MS08]. It defines non-redundant results as the minimal subset of subspace clusters to approximately compute the support of any other subspace cluster (assuming uniform distribution otherwise). This statistical approach, however, is based on the assumption of uniform distribution inside a cluster. Moreover, the redundancy model is limited to the fixed cluster definition.

Projected clustering. The paradigm of projected clustering was introduced by the PROCLUS method [AWY+99]. Inspired by traditional partitioning clustering methods like k-Means, projected clustering [AWY+99, MSE06, YCN05, BKKK04, YM03, PJAM02] partitions the data into disjoint clusters but simultaneously determines the most relevant dimensions for each group. Thus, projected clustering assigns each object to at most a single cluster. A partitioning of the data into projected clusters can be regarded as extreme redundancy elimination. Projected clustering results in a manageable number of clusters, but is not able to detect overlapping clusters. By limiting the clustering model to handle only disjoint clusters several meaningful clusters are only detected in parts or they are completely lost. In many application scenarios this is a general drawback, which is not acceptable for clustering in subspace projections.

As we show in our evaluations all of the existing approaches fail to detect all but only the hidden subspace clusters in a high dimensional database.

Contributions of the RESCU model. Our goal is to derive a novel model for non-redundant subspace clusters that takes a global look at overlapping clusters. The aim is to find all but only relevant clusters by optimizing the overall clustering result. The main contributions of our work are:

- Detection of all interesting clusters, but without the overwhelming result size of subspace clustering.
- Detection of only non-redundant clusters, but without the strict partitioning of projected clustering.

To achieve these objectives, we introduce a new global relevance model for sub-

space clustering. We combine an interestingness function for subspace clusters with a coverage criterion for an overall redundancy removal. By including the most interesting clusters and excluding redundant clusters, our result set contains all but only relevant subspace clusters. Any object can be part of multiple clusters, allowing overlapping subspace clusters. Furthermore, our relevance model is neither restricted to one cluster definition nor based on a fixed interestingness rating. These two aspects are typically application dependent and can be adapted by the instantiation of our model. Considering computation complexity, we prove that our relevant subspace clustering is NP-hard. Thus, for an efficient computation of our relevance model we propose an heuristic algorithm generating possible cluster candidates in best-first order according to their relevance.

2.2 Relevant Subspace Clustering

In this section, we introduce our relevant subspace clustering definition. Our model is independent of a certain subspace cluster definition, as e.g. density-based clusters [KKK04, AKMS07], but we simply assume that the set $All = \{C_1 \dots C_n\}$ of all subspace clusters that fulfill the selected definition is given. Note: As precomputation of All is usually too expensive we propose an heuristic algorithm using an on-demand cluster generation derived from our relevance model in Section 2.3. Most of the clusters in All do not provide any knowledge for the user, thus, our relevance model defines which of these clusters to output (cf. Fig. 2.1). In our model we utilize a global view for redundancy elimination, which considers the whole set All at the same time. Overall, we aim at reducing the output to only the *relevant* subspace clustering $Res \subseteq All$. The remaining clusters overwhelm the user, thus hinder the analysis and are removed.

Two aspects are important for relevance. One is the *interestingness* of a cluster itself. The interestingness evaluates a cluster locally via its properties like dimensionality or size. For example, clusters in located in one-dimensional subspaces might not be interesting. Interestingness is often user or application dependent and therefore it should be adaptable. A detailed discussion of our flexible interestingness model is given in Section 2.2.1.

Another aspect is the *redundancy* of clusters. Redundancy means that this cluster does not contribute to new knowledge with respect to other clusters (for example if another cluster with similar properties exists). Consequently, a redundant

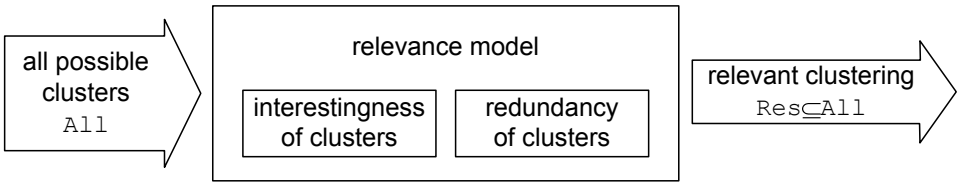


Figure 2.1: Components of a relevance model

cluster should not be reported. The redundancy is discussed in Section 2.2.2.

Please note that the two aspects *redundancy* and *interestingness* are not entirely independent in our model but they are considered simultaneously. We include the most interesting clusters but exclude redundant clusters of low interest. Thus, the user specified interest is also used in the redundancy elimination. Given the choice between two clusters with similar properties, the less interesting one should be marked as redundant. Overall, our optimization excludes both clusters with low interest and clusters that cover too few new objects (not yet covered by other clusters). This interleaved handling of interest and coverage of new objects is the key property of our optimization: a relevant cluster is interesting but not redundant.

Our overall relevance model is described in Section 2.2.3. We prove important complexity results in Section 2.2.4, discuss the parameters of our approach in Section 2.2.5 and conclude in Section 2.2.6 with an instantiation of our model.

2.2.1 Interestingness of a cluster

In this section, we describe the flexible interestingness model of RESCU to quantify the interestingness of a single cluster. Figure 2.2 gives an example of clusters in dimensions 1 and 2 (left), and 3 and 4 (right), respectively. Objects in both illustrations are represented using the same symbol. For example, assume that we deem clusters with more dimensions more interesting. The two-dimensional clusters C_1 and C_2 are then favored over the one-dimensional clusters $C_3 - C_6$. The number of objects, the diameter or the density of a cluster are other typical choices of interestingness. Besides these user driven specifications, further measures using statistical properties of the data might also be applicable and are widely used in other research areas such as frequent itemset mining [GH06, ST96]. By analyzing the data's distribution, such measures provide more objective criteria to evaluate the interestingness of a pattern and can be integrated in our model. However, as we focus more on the general perspective and the interleaved usage of interest in

our relevance model, we select a simple user-dependent interestingness measure based on the size and/or dimensionality of a cluster (cf. Section 2.2.6).

For the general perspective, let us discuss how interestingness is currently used (or not used) in subspace and projected clustering methods. Interestingness is not handled explicitly in existing projected clustering approaches. Since overlapping clusters are not permitted, the most interesting clusters, e.g. C_1 and C_2 in Figure 2.2, may not be found. They both contain some objects that occur in the other cluster as well, and are therefore mutually exclusive. As a consequence, potentially interesting clusters are missed by the occurrence of *other* interesting clusters. Another problem is the handling of outliers only in a post-processing step. Since the interestingness of clusters, e.g. their compactness, is calculated before removal of outliers, misleading values can occur.

Subspace clustering algorithms realize only a very limited interestingness calculation. Each set of points and dimensions that fulfills the cluster definition (e.g. exceeding a density threshold) is equally interesting. Thus, there is no specification of different degrees of interest in subspace clustering. It is a binary decision "cluster" or "no cluster". The user is not able to control what kind of clusters he is more interested in. After detecting all subspace clusters, no additional selection of the most interesting clusters out of the huge result set is performed.

Finally, the calculation of the interestingness in both models is usually fixed (or at least limited to the parameters introduced by the methods' developers) and cannot be modified by the user. A general and flexible interest specification providing the ability to instantiate the model by the application dependent needs is not provided.

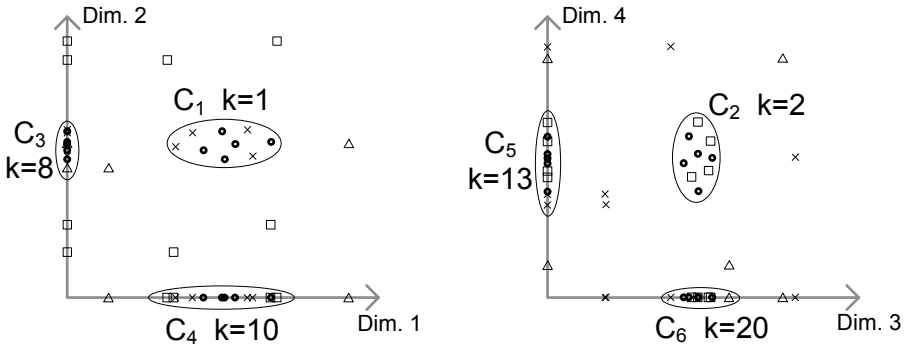


Figure 2.2: Interestingness by cost values

Our model overcomes these problems. We assign an (un-)interestingness value to each cluster. It is a local rating, i.e. other clusters do not influence this measure. Furthermore, our cluster definition (cf. Sec. 2.2.6) accounts for outliers. Thus, misleading values do not occur.

As a general and flexible framework, we model the (un-)interestingness via a cost function k. Intuitively, uninteresting clusters should get a high cost value since the benefit of analyzing these clusters during the knowledge discovery process is low but they only lead to high cost since, e.g., human resources or storage capacities are wasted for the analysis. Accordingly, small cost values denote interesting clusters. For any cluster C, given by the set of its objects O and the corresponding dimensions S, we define the cost function as follows:

Definition 2.1 *Cost function for subspace clusters*
Let $\mathcal{P}(DB)$ be the power set of all database objects and $\mathcal{P}(Dim)$ the power set of the dimensions. A cost function

$$k : \mathcal{P}(DB) \times \mathcal{P}(Dim) \to \mathbb{R}$$

assigns cost $k(O, S)$ to the subspace cluster $C = (O, S)$.

Please note that the domain of the cost function is not modeled by $\mathcal{P}(DB \times Dim)$. In this case, a single cluster C could contain objects that belong to different subspaces. For traditional subspace clustering, however, each object $o \in O$ of a given subspace cluster $C = (O, S)$ is associated with the same subspace S.

Exemplary cost values are depicted in Figure 2.2, where the one-dimensional cluster C_4 gets a cost value of $k = 10$ while the more interesting two-dimensional cluster C_1 gets a lower cost value of $k = 1$. By defining a cost function, we can account for several aspects of a subspace cluster, like the dimensionality or the density. Changes to the cost function yield an easy adaptation of the model. This flexibility is not achieved by other approaches. By first calculating the interesting values individually and allowing overlap we find higher quality clusterings. In Figure 2.2 we may select both C_1 and C_2.

For computational reasons, we assume cost functions which assign a strictly positive value to all subspace clusters, i.e. given a set $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$ of clusters, the function fulfills $k(O_i, S_i) > 0$ for all $(O_i, S_i) \in M$. To mine the most interesting clusters we minimize the overall cost. However, to also maximize

the number of objects covered by a clustering, we additionally take coverage into account. We formalize coverage as follows:

Definition 2.2 *Coverage of a clustering*
Given a clustering $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$*, the coverage of* M *is defined as follows:* $Cov(M) = \bigcup_{i=1}^{n} O_i$

The coverage of a clustering M is the union of the objects in all selected clusters. We now define the overall relative cost of a clustering as the sum of the individual cost values normalized by the number of covered objects.

Definition 2.3 *Overall relative cost of a clustering*
Let $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$ *be a clustering and* k *a cost function for subspace clusters. We define the overall relative cost of* M *as:*

$$RK(M) = \frac{K(M)}{|Cov(M)|} \text{ with } K(M) = \sum_{(O_i,S_i)\in M} k(O_i, S_i)$$

The smaller the overall relative cost $RK(M)$, the more interesting is the clustering M per covered object. Thus, we trade off the total cost of a clustering against its coverage. We achieve a high coverage and at the same time small total cost.

2.2.2 Redundancy of a cluster

The second aspect of relevance is non-redundancy. A large cluster C and all of its lower dimensional projections could be assigned low cost values if interestingness is based on size. Selecting all projections along with C based on interestingness alone leads to a poor overall result. One gets very many redundant clusters, while C would be sufficient.

Existing projected and subspace clustering algorithms do not address redundancy handling adequately. Projected clustering simply forces results to be non-redundant by assigning each object to a single cluster at the cost of missing overlapping clusters. Subspace clustering algorithms, in contrast, either use no or a mere local approach to check the redundancy. Such an approach compares only two clusters [AKMS07, AKMS08a, AKMS08b]. If the clusters cover nearly the same objects, one of them is redundant. The problem of this local approach is illustrated

in Figure 2.3. Obviously, in both subfigures, the cluster C_2 is redundant because it is induced by the other clusters C_1, resp. C_{1a}, C_{1b}. A local approach could identify the redundancy in the left figure. Cluster C_2 is redundant, as it covers C_1 and only a few additional objects. In the right figure, the fraction of points shared by C_{1a} and C_2 as well as by C_{1b} and C_2 is small, and the cluster C_2 is misleadingly classified as non-redundant. This mistake is the result of the local view on redundancy, i.e. for each check only a pairwise comparison of clusters is performed.

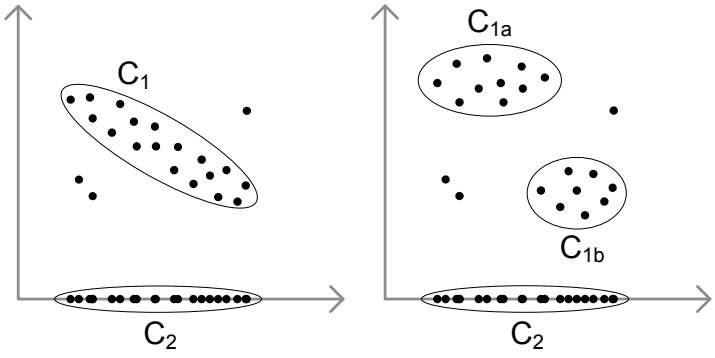


Figure 2.3: Local and global redundant clusters

We use a global view for the redundancy checks, i.e. we use all clusters at the same time to judge the redundancy of another cluster. This approach results in more accurate decisions. While the interestingness is a local measure based on the cluster itself, the redundancy takes other clusters into account.

As one can see from the above example, the redundancy of a cluster is linked to the coverage of objects. If a set of clusters shares many objects with a cluster C, C is a redundant cluster. In other words: A cluster is redundant if it does not cover many new objects. The cluster C_2 in Figure 2.3(b) is redundant, because with respect to the two other clusters only a few new objects are covered. The same holds for the cluster C_2 in Figure 2.3(a). The fact that we consider all clusters for the redundancy checks yields a global redundancy model. Thereby we identify in both subfigures the cluster C_2 as redundant.

Basic Set-Cover approach If we select the *minimal* number of clusters such that all objects are covered, we can realize a global redundancy model. The output of clusters that only contain already covered objects is prevented. This check is

performed with respect to all other clusters in the result set. At the same time, we identify multiple overlapping clusters in the data because all objects have to be covered.

This novel way of subspace clustering can be considered as an instance of the Set-Cover problem [GJ79]. Given several finite sets, Set-Cover seeks for the minimal number of sets that cover the whole population. In the clustering context, we want to find the minimal number of clusters such that all objects are covered.

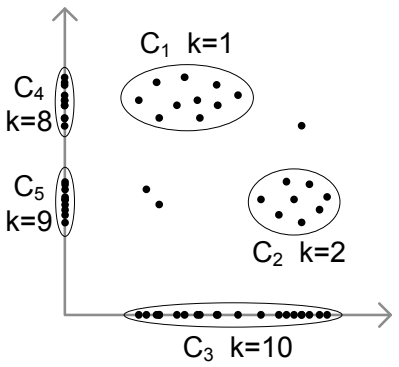
However the direct application of Set-Cover to our task is not possible.

Problem 1: Simply choosing the minimal number of clusters means that usually low dimensional clusters, which tend to contain more objects, are preferred over high dimensional clusters. This preference usually conflicts with the user's notion of interestingness. Instead of choosing the minimal number of clusters, we determine the clustering with minimal relative cost (cf. Def. 2.3). This setup takes the desired interestingness notion into account. In Figure 2.3(b), for example, we would choose the two two-dimensional clusters instead of the one one-dimensional cluster. We thus use an extension of the Set-Cover problem to the Weighted-Set-Cover problem [Chv79].

Problem 2: Covering all objects by clusters is not always a meaningful solution, as some databases contain outliers that do not fit to any cluster. The Set-Cover problem enforces a complete cover of all objects and potentially finds no solution in this case. Or all objects are covered by uninteresting clusters only. For example, if the outliers are only contained in the one-dimensional clusters, the Set-Cover problem enforces choosing these uninteresting ones. An example is depicted in Figure 2.3(b), as one has to select C_2 to get a complete coverage. We therefore propose a generalization of the Set-Cover problem to cope with outliers.

Gain-based redundancy We solve these problems by a gain-based extension of the Set-Cover problem. The basic idea is to measure the gain of a new cluster if we add it to a known clustering. In other words, we have to answer the question: Is it worthwhile to take the "more complex" clustering?

Let us consider Figure 2.4 and assume the clusters C_1 and C_2 to be selected. Intuitively, the cluster C_3 is redundant with respect to the selected ones, so its gain should be small. Two important aspects contribute to this fact. First, the cluster C_3 covers only a few new objects, i.e. many objects are already contained in other clusters. These new objects, covered by the cluster $C_3 = (O, S)$, can be calculated



$$clus_gain(C_3, \{C_1, C_2\}) = \frac{3}{10} = 0.3$$
$$clus_gain(C_4, \{C_1, C_2\}) = \frac{1}{8} = 0.13$$
$$clus_gain(C_2, \{C_1\}) = \frac{8}{2} = 4$$
$$clus_gain(C_1, \{C_2\}) = \frac{10}{1} = 10$$
$$\frac{K(\{C_1,C_2\})}{|Cov(\{C_1,C_2\})|} = \frac{3}{18} = 0.17$$
$$\frac{K(\{C_4,C_5\})}{|Cov(\{C_4,C_5\})|} = \frac{17}{21} = 0.81$$

Figure 2.4: Relevant clustering

via the residual set $O\backslash Cov(\{C_1, C_2\})$. And second, the cost of the cluster is very high, i.e. the cluster is not interesting. High cost $k(O, S)$ should correspond to a low gain. Overall we take the ratio of these two measures. Cluster gain thus is the additional coverage in relation to its cost.

Definition 2.4 *Cluster gain*
Given a cluster $C = (O, S)$, a clustering M and a cost function k for $M \cup \{C\}$. The cluster gain of C with respect to M is:

$$clus_gain(C, M) = \frac{|O\backslash Cov(M)|}{k(O, S)}$$

We identify a cluster as redundant (with respect to a given clustering) if its cluster gain is smaller than a user-defined threshold Δ. In Figure 2.4 the cluster gain of C_3 (with respect to $\{C_1, C_2\}$) is only 0.3 as C_3 covers 3 new objects and $k(C_3) = 10$. Instead, C_2 has a higher cluster gain of 4 (with respect to $\{C_1\}$).
Consistent with this idea, a clustering M is redundancy-free if *all* clusters from M exceed the selected minimal cluster gain Δ. The gain is measured with respect to the remaining clusters from M so that each cluster adds sufficient information to the overall clustering.

Definition 2.5 *Redundancy-free clustering*
Given a clustering $M \subseteq All$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. The clustering M is redundancy-free, iff

$$\forall C \in M : clus_gain(C, M\backslash\{C\}) > \Delta$$

In this way we achieve a global view for the redundancy checks. Each subspace cluster C has to compete with all remaining clusters $M \backslash \{C\}$ in the result set at the same time.
As one can see, the clustering $\{C_1, C_2\}$ in Figure 2.4 is redundancy-free while the clusterings $\{C_1, C_2, C_3\}$ or $\{C_1, C_2, C_4\}$ contain redundant information (assuming $\Delta = 0.5$). Definition 2.5 gives us the basis to avoid redundancy in our clustering. However, if we remove clusters from a redundancy-free clustering, this property still holds for the smaller clustering. The removal of information, i.e. clusters, cannot generate redundancy. This means that also the empty clustering is redundancy-free. That is obviously not desired by the user. To obtain a relevant clustering, it should be redundancy-free but at the same time cover as many objects as possible.

2.2.3 Overall relevance of a clustering

To enforce the selection of clusters we introduce the property of concept-covering. A clustering M does *not* satisfy this property as long as clusters exist that have a high gain and are not in M. M is not the result because further interesting concepts, represented by sets of clusters, can be covered. A clustering is concept-covering, and hence the coverage is sufficient, if all remaining clusters have a small cluster gain. We formalize this by:

Definition 2.6 *Concept-covering clustering*
Given a clustering $M \subseteq All$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. The clustering M is concept-covering, iff

$$\forall C \in All \backslash M : clus_gain(C, M) \leq \Delta$$

Intuitively, a clustering is concept-covering if adding *any* new cluster *always* results in redundancy. A concept-covering clustering in Figure 2.4 is $\{C_1, C_2\}$. Clustering $\{C_1\}$ is not concept-covering because we could add C_2 without introducing redundancy.

The property of concept-covering clusterings is a relaxation of complete coverage. It solves the problem of enforcing a complete coverage as in the Set-Cover problem (cf. Problem 2).

RESCU: Relevant subspace clustering Our RESCU model demands a relevant clustering to be redundancy-free and concept-covering. However, as the example in Figure 2.4 illustrates, several clusterings could fulfill both properties, e.g. $\{C_1, C_2\}$ or $\{C_4, C_5\}$. To select the most interesting clustering, we additionally compare their relative cost (cf. Def. 2.3). Our relative cost function (cost per covered object) ensures finding an optimal clustering with both interesting clusters and high coverage. The relative cost for the clustering $\{C_1, C_2\}$ in Figure 2.4 is just 0.17, but 0.81 for $\{C_4, C_5\}$. This is formalized by:

Definition 2.7 *Relevant subspace clustering (RESCU)*
Given a clustering $Res \subseteq All$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. Res is relevant, if and only if

- *Res is redundancy-free (Def. 2.5) and concept-covering (Def. 2.6) AND*
- *Res has minimal relative cost, i.e. $RK(Res) \leq RK(N)$ for all redundancy-free and concept-covering clusterings $N \subseteq All$*

The relevant clustering in our previous example is thus $\{C_1, C_2\}$. Also for the example in Figure 2.2 the relevant clustering is $\{C_1, C_2\}$ even though the two clusters share some objects. This illustrates that handling of overlapping clusters is possible in our model.

The flexibility of our model additionally provides the possibility of classifying other clusterings as relevant by changing the interestingness criterion. If we adapt the cost function so that C_1 and C_2 in Fig. 2.4 get higher cost values, the clustering $\{C_4, C_5\}$ could become relevant. Thus our model enables the user to control the output as desired.

2.2.4 Complexity analysis

In the following we show that calculating a RESCU clustering is an NP-hard problem. We prove this by giving a polynomial reduction of the Weighted-Set-Cover problem, which is a NP-complete problem, to our RESCU model, i.e. we prove Weighted-Set-Cover $\leq_p RESCU$. The Weighted-Set-Cover problem seeks those non-empty sets that together have minimal weights and fully cover all objects. For this reduction we need to map the input of the Weighted-Set-Cover problem to an

input of RESCU and show that the resulting relevant subspace clustering is a valid solution for the Weighted-Set-Cover problem.

Theorem 2.1 *Computing RESCU (Def. 2.7) is NP-hard.*

Proof 2.1
We show that Weighted-Set-Cover $\leq_p RESCU$.
Input mapping: We map the input of the Weighted-Set-Cover (objects, sets, weight function) to an input for RESCU (database DB*, possible clusters* All*, cost function* k*). We map the assumption of the Set-Cover (complete coverage exists) to (*Δ*=0) in RESCU. We now have to show that relevant subspace clustering (cf. Def. 2.7) corresponds to a solution of the Weighted-Set-Cover problem.*
RESCU generates a valid solution for Set-Cover:

(1) Every chosen set contributes at least one object to the overall coverage:
A relevant clustering Res *is non-redundant (Def. 2.5):*
$\Leftrightarrow \forall\, C \in Res : clus_gain(C, Res \backslash \{C\}) = \frac{|O \backslash Cov(Res \backslash \{C\})|}{k(O,S)} > 0$
Thus, all sets contribute at least one object
$\forall\, C \in Res : O = Cov(\{C\}) \not\subseteq Cov(Res \backslash \{C\})$.

(2) For a Weighted-Set-Cover all objects have to be covered by at least one set:
A relevant clustering Res *fulfills the concept-covering property (Def. 2.6):*
$\Leftrightarrow \forall\, C \in All \backslash Res : clus_gain(C, Res) = \frac{|O \backslash Cov(Res)|}{k(O,S)} \leq 0$
$\Leftrightarrow \nexists\, C \in All \backslash Res :\ |O \backslash Cov(Res)| > 0$
Thus, all objects are covered: $Cov(Res) = DB$.

(3) The sum of weights is minimal for the chosen sets:
A relevant clustering Res *has minimal relative costs (Def. 2.7):*
For all non-redundant and concept-covering clusterings
$N \subseteq All$*:* $RK(Res) \leq RK(N) \Leftrightarrow \frac{K(Res)}{|Cov(Res)|} \leq \frac{K(N)}{|Cov(N)|}$
From $Cov(Res) = Cov(N) = DB$ *we have:* $K(Res)$ *is minimal.*

(1) $\wedge$ *(2)* $\wedge$ *(3)* $\Rightarrow$ Res *is a valid Set-Cover solution.* $\Rightarrow$ *RESCU is NP-hard.* □

As we have proven, RESCU is a generalization of the Set-Cover problem and thus NP-hard. Furthermore, it has two advantages for detection of relevant clusterings. First, we can handle outliers so that we can find a solution even if a complete coverage is not possible. And second, RESCU incorporates clustering properties and thus mines the most relevant clusters instead of simply covering the data.

2.2.5 Discussion of Parameters

Our RESCU model requires just a single parameter: Δ. The parameter specifies the minimal gain each cluster of the result has to exceed. The parameter can be selected in the range $[0, max_g)$ where $max_g = \max_{(O_i,S_i)\in All} \frac{|O_i|}{k(O_i,S_i)}$ is the maximal gain a single cluster can achieve. For $\Delta \geq max_g$, the result will be empty. By lowering the parameter, redundancy is evaluated less strict, i.e. clusters are allowed to overlap to a higher degree. For $\Delta \to 0$ arbitrary overlap between clusters is possible as far as a cluster is not completely covered by other clusters.

Furthermore, the parameter Δ intuitively controls the maximal relative cost the user is willing to accept for the final result. It holds that the relative cost of an optimal RESCU clustering $Res \subseteq All$ is upper bounded by $\frac{|DB|}{\Delta}$. Since $|Cov(Res)| \geq \sum_{C_i\in Res} |O_i \backslash Cov(Res \backslash C_i)|$ and each cluster covers at least one additional object compared to the remaining clusters, we get $RK(Res) = \frac{K(Res)}{|Cov(Res)|} \leq \frac{\sum_{C_i\in Res} k(O_i,S_i)}{\sum_{C_i\in Res} |O_i\backslash Cov(Res\backslash C_i)|} \overset{(*)}{\leq} \sum_{C_i\in Res} \frac{k(O_i,S_i)}{|O_i\backslash Cov(Res\backslash C_i)|} < \sum_{C_i\in Res} \frac{1}{\Delta} = \frac{|Res|}{\Delta} \leq \frac{|DB|}{\Delta}$. The inequality $(*)$ is true as shown by the following theorem.

Theorem 2.2 (Sum inequality) *Let $n_1, \ldots, n_m$ and $d_1, \ldots, d_m$ be two sequences with $n_i \in \mathbb{R}_0^+$, $d_i \in \mathbb{R}^+$ for all $i \in \{1, \ldots, m\}$. It holds: $\sum_{i=1}^m \frac{n_i}{d_i} \geq \frac{\sum_{i=1}^m n_i}{\sum_{i=1}^m d_i}$*

Proof 2.2 *Let $L = \max_{i\in\{1\ldots,m\}}\{\frac{n_i}{d_i}\}$. We get $\sum_{i=1}^m \frac{n_i}{d_i} \overset{(1)}{\geq} L = L \cdot \frac{\sum_{i=1}^m d_i}{\sum_{i=1}^m d_i} = \frac{\sum_{i=1}^m L\cdot d_i}{\sum_{i=1}^m d_i} \overset{(2)}{\geq} \frac{\sum_{i=1}^m n_i}{\sum_{i=1}^m d_i}$ Inequality (1) holds, since each term $\frac{n_i}{d_i}$ is non-negative. Inequality (2) holds, since for all i we have $L \geq \frac{n_i}{d_i} \Leftrightarrow L \cdot d_i \geq n_i$ and d_i is positive.* □

Thus, overall our model ensures the bound $RK(Res) < \frac{|DB|}{\Delta}$ for an optimal RESCU clustering $Res \subseteq All$. This result can, for example, be used to find appropriate parameter settings for Δ. For example, if the user selects a cost function that rates the dimensionality of clusters and if the user wants to ensure that the average dimensionality of the clusters in the final result exceeds a certain value, we can derive a value for Δ guaranteeing this property.

2.2.6 Instantiation of the model

By the flexibility of our model we can handle any definition of subspace clusters (as the input of the model) and cost functions (to rate the clusters). For a practical evaluation we instantiate these two aspects: for the cluster definition we use

the frequently used paradigm of density-based clusters, while the cost function is based on the dimensionality of the clusters.

Definition of subspace clusters. We use density-based clustering because it detects clusters of arbitrary shape and size even in noisy data [EKSX96]. The idea is to define clusters as dense areas separated by sparsely populated areas. In this way outliers are not part of the clusters within the same subspace, and misleading ratings are prevented. An object is considered dense if its neighborhood, i.e. an ε-distance region around it, is sufficiently populated. We follow the definition from [KKK04] with the modification that we adjust the ε-range according to the subspace dimensionality. Thereby we account for increasing distances between the objects in higher dimensional spaces. Thus, a subspace cluster is defined by

Definition 2.8 *Given a set of dimensions $Dim = \{1, \ldots, D\}$, a database of objects $DB \subset \mathbb{R}^D$, and an adaptive neighborhood range function $\varepsilon(d) : \mathbb{N} \to \mathbb{R}^+$, a subspace cluster $C = (O, S)$ with $O \subseteq DB$ and $S \subseteq Dim$ fulfills the following properties:*

(1) high density: $\forall o \in O : |\{p \in DB \mid dist_S(o, p) \leq \varepsilon(|S|)\}| \geq \tau$

(2) density connected: $\forall o, p \in O : \exists o_1, \ldots, o_k \in O : o_1 = o \wedge o_k = p \wedge \forall i = 1, \ldots, k-1 : dist_S(o_i, o_{i+1}) \leq \varepsilon(|S|)$

(3) maximality: $\neg\exists O' \supset O$ *fulfilling (1) and (2)*

where $dist_S(o, p) = \sqrt{\sum_{i \in S} |o[i] - p[i]|^2}$ *is the Euclidean distance restricted to the subspace S.*

While traditional density-based clustering methods and many subspace clustering approaches use a constant neighborhood range, we use a monotonically increasing function $\varepsilon(d)$. Since the distance between objects increases with increasing subspace cardinality, using a constant density criterion is problematic: By using a small ε (or large τ) we cannot expected any high dimensional clusters; most objects have a distance larger than ε. By using a large ε (small τ), however, the low dimensional clusters are no longer meaningful since they group nearly all objects in a single cluster; most objects have a distance smaller than ε. A density criterion that is identical for each subspace is not an appropriate choice for subspace clustering. We have to use an adaptive density measure.

In general, the density calculation can be adapted to the subspace cardinality by modifying either ε, i.e. the neighborhood range, or τ, i.e. the minimal number

of objects in the neighborhood. While an adaption of ε directly antagonizes the increasing distances, τ acts only indirectly (increasing distances $\rightarrow$ fewer objects in ε-neighborhood $\rightarrow$ smaller τ).

The indirect adaptation based on τ, as for example performed by [AKMS07, SZ04], is problematic since the value of τ quickly converges to 0, i.e. even objects with just a single neighbor are regarded as dense areas. To prevent such misleading clusters the methods of [AKMS07, SZ04] introduced an artificial constant that bounds the value of τ to a minimal value. Thus, in our method (and similar in [KKRW05]) an adaption of ε is performed to expect non-empty neighborhoods also in higher-dimensional subspaces. Formally, we adapt the value of $\varepsilon(d)$ in a subspace with dimensionality d by

$$\varepsilon(d) = \left[\frac{8 \cdot n}{3 \cdot \sqrt{\pi}}\right]^{\frac{1}{5}} \cdot \varepsilon_1 \cdot \left[\frac{d+2}{4 \cdot n} \cdot \Gamma(1 + \frac{d}{2})\right]^{\frac{1}{d+4}} \tag{2.1}$$

where ε_1 denotes the ε-range in the one-dimensional subspaces, n the database size and Γ the gamma function.

Our adaptive function is derived from a statistically optimal choice of ε provided by kernel bandwidth estimators [Sil86]. Clustering based on kernel density estimation [AKMS07, HK98], is an extension of the traditional density-based clustering paradigm as, e.g., performed by [EKSX96]. More precisely, traditional techniques as [EKSX96] use a constant kernel (also denoted as the uniform kernel K_U): all objects in the ε-neighborhood have the same importance, i.e. one just considers the number of objects in a certain neighborhood to determine the density. Using this observation, we can determine the function $\varepsilon(d)$ based on existing methods that estimate the bandwidths of kernels. That is, if $h_d(K)$ provides a bandwidth estimation for the d-dimensional kernel K, a reasonable neighborhood range function should fulfill the property $\varepsilon(d) \sim h_d(K)$.

According to [Sil86], the optimal kernel bandwidth $h_d(K)$, given a fixed dimensionality d and the kernel function K, can be estimated based on

$$h_d(K) := \left[d \cdot \beta_d(K) \cdot \alpha_d(K)^{-2} \cdot (2\sqrt{\pi})^d (\frac{1}{2}d + \frac{1}{4}d^2)^{-1} \cdot \frac{1}{n}\right]^{1/(d+4)} \tag{2.2}$$

$$\text{with} \qquad \alpha_d(K) := \int_{x \in \mathbb{R}^d} x_1^2 \cdot K(x)\, dx \qquad \beta_d(K) := \int_{x \in \mathbb{R}^d} K(x)^2\, dx$$

For the above mentioned uniform kernel K_U, we get the results $\alpha_d(K_U) = \frac{1}{d+2}$ and $\beta_d(K_U) = \frac{\Gamma(1+\frac{d}{2})}{\pi^{d/2}}$ leading to the simplified formula

$$h_d(K_U) = \left[\frac{d+2}{4 \cdot n} \cdot \Gamma(1+\frac{d}{2})\right]^{\frac{1}{d+4}} \cdot 2 \tag{2.3}$$

By using an initial choice of ε_1 for the one-dimensional subspaces and by scaling this value with increasing dimensionality, i.e.

$$\varepsilon(d) \sim h_d(K_U) \qquad \text{and} \qquad \varepsilon(1) = \varepsilon_1$$

should hold, we can easily derive Equation 2.1. Overall, the parameter ε_1 can be used to adapt our model to application dependent properties while the scaling to different subspaces is done automatically.

Definition of cost functions. The cost function used in our experiments is selected as $k(O,S) = \frac{1}{|S|^\beta}$ with $\beta \geq 0$. Higher dimensional clusters get lower cost values and are therefore more interesting than lower dimensional clusters. Preference for higher dimensional clusters is also used in [AKMS08a]. Variation of β influences how much different subspace dimensionalities affect the cost value. A very high β-value implies that the result set mainly contains high dimensional clusters, whereas a low value tends to lead to lower dimensional clusters. If all cost values are nearly identical, low dimensional clusters are usually preferred because these clusters generally cover more objects and so their cluster gain is higher. Overall, by selecting the cost function as above, the cluster gain accounts for the two major aspects of subspace clusters: their object cardinality as well as their subspace cardinality.

2.3 The RESCU Algorithm

There are two major challenges for the efficient computation of a relevant subspace clustering. First, as we have shown in Section 2.2.4, the global optimization of the clustering is an NP-hard problem. And second, the assumed input set All is too large and its enumeration may be difficult or even impossible. Thus, we assume that there exists no efficient and especially no scalable solution. However, we propose an algorithm computing an approximate solution that tackles both of these challenges based on the following contributions:

- Greedy selection of clusters and relevance update.
- On-demand generation and ranking of subspace clusters according to their cluster gain.

A naive approach would compute all possible subspace clusters (All is exponential w.r.t. the number of dimensions) and then chooses an optimal subset of clusters (exponential in the number of cluster candidates). We use an approach that generates promising clusters on-demand and ranks them according to their relevance information. Thus, we avoid an expensive pre-computation of all possible subspace clusters. With our greedy approach we relax the optimization by choosing in each step the most promising cluster available. Please note that a new cluster changes the overall coverage of the data, and it changes the relevance of remaining clusters. Updating the relevance is thus essential for concept-covering. The overall pseudo-code of our RESCU approach is illustrated in Algorithm 2.1.

Greedy Selection of Clusters and Relevance Update
Our greedy approach iteratively includes the best cluster so far. Such an approximation idea is known for other NP-hard problems like Set-Cover [GJ79, Chv79]. For our relevant subspace clustering, this basic idea of greedy processing leads to efficient computation but also high quality subspace clustering results by choosing in each step the most relevant cluster according to our cluster gain definition.

By iteratively picking clusters, we relax two parts of Definition 2.7. First, instead of checking the global redundancy (Def. 2.5) we compute an approximate solution, as in each step i the relevance of a new cluster C_i is checked only w.r.t. clusters $M_i = \{C_1, \ldots, C_{i-1}\}$ already chosen in the previous $i-1$ steps. The cluster C_i is non-redundant w.r.t. M_i. Second, we choose the cluster with the highest $clus_gain$ in order to have the most interesting clusters in the result and to approximate the minimal relative cost of the relevant clustering. Formally, in step i, we insert C_i into the current result set M_i iff

(1) C_i is a non-redundant cluster w.r.t. M_i: $clus_gain(C_i, M_i) > \Delta$

(2) C_i is the most interesting cluster in step i: $\forall C \in All \backslash M_i : clus_gain(C_i, M_i) \geq clus_gain(C, M_i)$

Our relaxation leads to an efficient processing. It computes a chain of most relevant clusters C_i yielding an approximate solution for our RESCU model. It

Algorithm 2.1: Overall processing scheme of the RESCU algorithm

```
1  Result = ∅ // current result set
2  queue = ∅ // ranking of clusters, sorted in descending order by cluster gain
3  initialize queue with a (small) set of clusters
4  while queue ≠ ∅ do
5      remove first cluster C from queue (highest cluster gain)
6      if C was not used for on-demand generation then
7          generate clusters based on C and queue → set of clusters 𝒞
8          add clusters of 𝒞 and C to queue
9      else
10         if cluster_gain(C, Result) > Δ then
11             add C to Result
12             update ranking
13         else
14             break; // Result is concept-covering
```

is a best-first method as in each step the cluster with the highest gain is selected (Algorithm 2.1, line 5). The greedy processing terminates if no more relevant clusters are available in the residual set of clusters according to the concept-covering property (cf. Def. 2.6). This is illustrated in lines 10-14 of Algorithm 2.1.

The set of resulting clusters directly influences the cluster gain of the next cluster to be chosen. As the new cluster C_i changes the overall coverage of objects, i.e. $Cov(M_{i+1}) = Cov(M_i \cup \{C_i\}) \supset Cov(M_i)$, we have to adjust the relevance of all remaining cluster candidates, i.e. the cluster gain (Definition 2.4) of all candidates $C \in All \backslash M_{i+1}$ is updated. Our relevance update (Algorithm 2.1, line 12) decreases the gain of redundant clusters that are already covered by C_i. Consequently, other clusters not yet covered have a relatively higher likelihood of being chosen in the following iteration.

In our example given in Figure 2.4 we first choose C_1 as the first relevant subspace cluster with maximal cluster gain $clus_gain(C_1, \{\}) = 10$. Intuitively, our algorithm chooses the clusters according to the cluster gain definition, which prefers higher dimensional clusters that contain objects not yet covered by other relevant clusters. Thus, in the second step we choose C_2 with the currently highest gain $clus_gain(C_2, \{C_1\}) = 4$, a 2d-cluster with not yet covered objects. Each selected cluster forces a relevance update and in our example, most of the objects in C_3, C_4, C_5 are now already covered. Thus, the next most relevant cluster is C_3 with an updated relevance of only 0.3 as it contributes only 3 objects to the overall

clustering. Assuming $\Delta = 0.5$, the algorithm has detected all and only the relevant clusters as any remaining cluster has a lower cluster gain.

On-Demand Generation and Ranking

For our greedy processing we maintain an up-to-date ranked list of subspace clusters with a high cluster gain (Algorithm 2.1, line 2). However, it would be computationally too expensive to compute all possible clusters All and then sort them according to their cluster gain. In contrast to such an exhaustive generation of all possible clusters, our approach computes candidates on-demand and reduces computation to only the most promising regions. As most of the exponentially many clusters in All are not interesting or they are redundant w.r.t. the final result set, they do not affect the global optimization. Hence, our on-demand candidate generation computes only the most promising cluster candidates according to their cluster gain.

For on-demand generation of clusters, we use the technique introduced in [MAGS11]. Given a set of already known clusters K, the technique of [MAGS11] selects a cluster $C = (O, S) \in K$ to generate a set $\mathcal{C}$ of higher dimensional clusters located in subspaces $S' \supset S$ that cover similar sets of objects. Unlike traditional exhaustive enumeration techniques, which are, e.g., based on depth-first search, the cardinality of the newly generated subspaces S' might be much higher than $|S|$, i.e. in each step the dimensionality of the clusters can increase by $x > 1$. The information which dimensions are reasonable candidates for an extension of $C = (O, S)$ is obtained by analyzing the set of known clusters K and their similarity w.r.t. C. Intuitively, this approach adapts and generalizes the merging step known from bottom-up Apriori-methods but it uses an approximate generation of high dimensional clusters to allow a high efficiency. A similar method was also introduced for itemset mining [ZYH+07].

In our method, the set of known clusters corresponds to the current ranking of the clusters, and according to [MAGS11] we initialize this set by analyzing the 2d subspace projections of the data (Algorithm 2.1, line 3). Based on this set, we pick the cluster C with the highest cluster gain to generate a set of novel clusters $\mathcal{C} = \{C_1, \ldots, C_k\}$ since such a cluster C is the most promising one to generate further clusters with a high gain (line 5,7). These novel cluster candidates $\mathcal{C}$ are inserted into our ranking based on their cluster gain (line 8). They thus increase the amount of known clusters that can be used for determining the final result

and for generating further clusters in later iterations. If the cluster with the highest cluster gain was already used for generating further candidates (line 9), we proceed with the previously described greedy processing, i.e. we test whether the cluster can be added to the final result.

Please note that due to relevance update, the positions of the new cluster candidates might be rearranged. Thus, at each time, the currently most relevant candidates are at the top of the ranking. By choosing the top candidate from the ranking, we focus in each step on the most promising cluster either to generate new candidates or to complete the result. Using this interleaved processing of on-demand generation and greedy selection, we ensure efficiency by generating only a small set of promising subspace cluster candidates as needed and selecting only the most promising (top ranked) cluster for processing.

2.4 Experimental Analysis

We compare RESCU with recent representatives of different high dimensional clustering paradigms: Grid-based subspace clustering CLIQUE [AGGR98] and its extension SCHISM [SZ04]; Density-based algorithms SUBCLU [KKK04], INSCY [AKMS08a] and FIRES [KKRW05]; Projected clustering PROCLUS [AWY+99] and statistical P3C [MSE06] and StatPC [MS08]. Note that we have optimized parameters for each algorithm on each data set. Furthermore, we provide supplementary material (executables, exact parameter settings and data sets used in our evaluation) for repeatability and comparison on our website. All implementations are in Java and experiments were run on Intel Core 2 Duo computers with 3 GHz and 2 GB main memory.

Benchmark data from the UCI archive [FA10] (also used in the evaluation study in [MGAS09]) is used to study performance on real data. In addition, we use 17-dimensional features as extracted in [AKMS08a] from sequence data in [KWX+06]. Besides the UCI 16-dimensional pendigits data, we use 32 and 48-dimensional variants by interpolation of the available polylines. As the true number of clusters is unknown for real data, accuracy w.r.t. class labels, as also used in [MSE06, AKMS08a, MAK+09], is measured. We employ synthetic data for validating that all generated clusters are identified and for scalability experiments. Following the method in [KKK04, AKMS08a] for overlapping density-based clusters in arbitrary subspaces, we generate data of different dimensionalities and

hide subspace clusters with a dimensionality of 50%, 60% and 80% of the data dimensionality.

We measure quality via the F1 value [MSE06, AKMS08a, MAK+09, WF05]. It is computed as the harmonic mean of recall ("are all clusters detected?") and precision ("are the clusters accurately/purely detected?"). On real data, the class label of a detected subspace cluster is its most frequent label. The F1 value of the entire clustering is the average of all clusters' F1 values. Additionally, we provide the commonly used accuracy of classifiers (e.g. C4.5 decision tree) built on the detected patterns [BZ07, MAK+09]. A high accuracy indicates that the subspace clustering is a good generalization of the underlying data distribution. By providing both measures on many data sets, we provide a thorough analysis for real world data.

2.4.1 Scalability

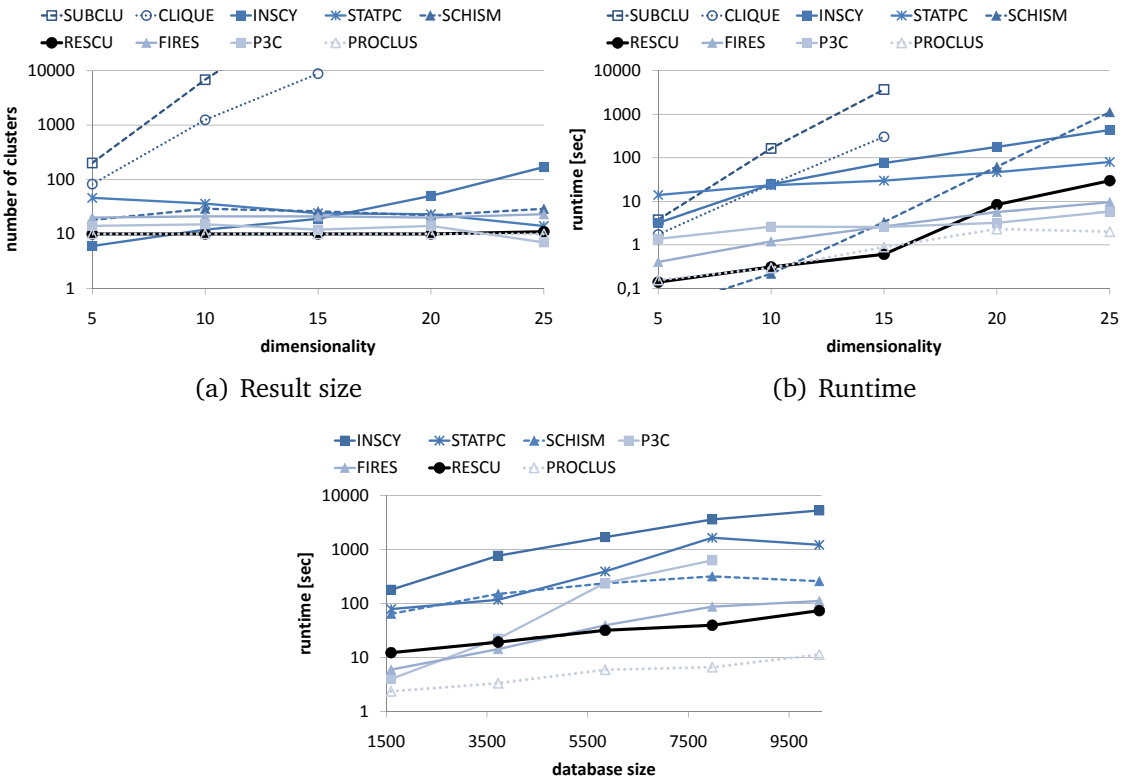
Our comparative study begins with an analysis of the result size and runtime (Fig. 2.5(a) & 2.5(b)) with respect to the dimensionality on a synthetic data set with 10 hidden clusters.

Redundancy maintenance: First, we want to show the result size of the algorithms CLIQUE and SUBCLU, which do not provide redundancy removal. They produce overwhelming result sets that are several orders of magnitude larger than the number of hidden clusters. They suffer from the fact that any cluster typically induces several very similar clusters in lower dimensional projections, especially as dimensionality increases. Our RESCU approach successfully detects only the 10 relevant subspace clusters.

Figure 2.5(b) illustrates the effect of size on the runtime of these approaches. (Please note the logarithmic scale.) Beyond 15 dimensions, the poor scalability of CLIQUE and SUBCLU due to their result size renders analysis infeasible on standard desktop PCs. In the following experiments, we do not include CLIQUE and SUBCLU due to their poor performance in terms of result size and runtimes. RESCU, as a representative for models with redundancy removal, clearly outperforms these algorithms.

In summary, redundancy removal is a key property to provide interpretable results and good scalability. Algorithms that maintain redundant clusters are not considered in the following experiments.

Redundancy removal: Compared to CLIQUE and SUBCLU the remaining algo-



(a) Result size

(b) Runtime

(c) Runtime

Figure 2.5: Scalability w.r.t. dimensionality and size of the database

rithms scale to higher dimensional data sets and show significantly smaller result sizes in the range from 6 to 169 as depicted in Figure 2.5(a). Please note that comparison with some algorithms that require the number of clusters as an input, like the projected clustering approach PROCLUS, does not reflect their performance in real application scenarios where this information is typically not available. Most of the algorithms output more than 10 clusters, however, they where not able to detect all of the 10 hidden clusters. RESCU is able to find all hidden clusters. Overall only our RESCU relevance model is able to find the hidden and, thus, relevant clusters. We further investigate the quality of the clustering result on real world data sets in Section 2.4.2.

The runtime of RESCU (Fig. 2.5(b)) is comparable to that of the other approaches. It is less affected by the dimensionality as it computes only the relevant subspace clusters on-demand and excludes most of the irrelevant clusters. Our relevance ranking of cluster candidates and the greedy processing ensures an overall

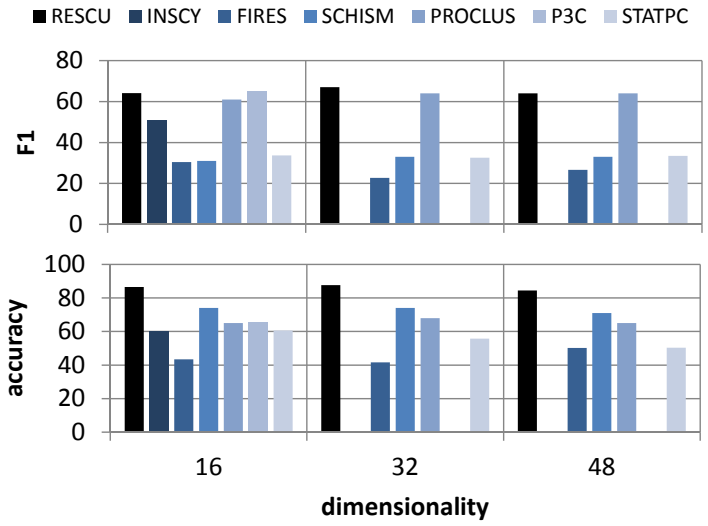


Figure 2.6: Quality on pendigits data (7494 objects; 16-48 dimensions)

efficient computation. Although some algorithms have smaller runtime, RESCU is still efficient and, as we believe, this aspect is compensated by the higher clustering quality of RESCU.

Our next experiment in Figure 2.5(c) shows the runtime of the algorithms with respect to the database size. Our RESCU approach scales well, whereas most of the competing algorithms show a greater increase in runtime. Overall the results from this experiment are comparable to the results in Fig. 2.5(b).

2.4.2 Quality on real world data

Our next experiment in Figure 2.6 evaluates F1 value and accuracy for the pendigits data. We vary the dimensionalities from 16 to 48. For F1 measure on the 16d data set we observe top quality results for RESCU, P3C and PROCLUS. Hence these algorithms find almost all and pure clusters. While P3C does not scale to the higher dimensional data sets, RESCU and PROCLUS reach again high qualities. Considering accuracy, our RESCU approach has the best quality results. SCHISM is second in accuracy, but has a significantly lower F1 value. In general, RESCU is the only approach that shows top results for both measures in all dimensionalities. Our novel model is able to detect the most interesting and non-redundant clusters in the data set.

It must be highlighted that the results of the two density-based approaches

	Glass (214; 9)				*Vowel (990; 10)*				*Diabetes (768; 8)*			
	F1		*Accuracy*		*F1*		*Accuracy*		*F1*		*Accuracy*	
RESCU	60	100%	62	100%	44	100%	64	96%	71	100%	69	100%
INSCY	56	93%	54	87%	37	84%	67	100%	58	82%	65	94%
FIRES	30	50%	49	79%	10	23%	12	18%	33	46%	65	94%
SCHISM	45	75%	49	79%	24	55%	53	79%	69	97%	69	100%
PROCLUS	39	65%	54	87%	32	73%	30	45%	44	62%	65	94%
P3C	17	28%	39	63%	8	18%	16	24%	44	62%	65	94%
STATPC	19	32%	47	76%	17	39%	47	70%	39	55%	64	93%

	Shape (160; 17)				*Liver-Dis. (345; 6)*				*Breast (198; 33)*			
	F1		*Accuracy*		F1		*Accuracy*		*F1*		*Accuracy*	
RESCU	60	100%	75	100%	62	97%	61	98%	67	100%	76	97%
INSCY	56	93%	61	81%	62	97%	59	95%	65	97%	70	90%
FIRES	56	93%	62	83%	50	78%	53	85%	46	69%	75	96%
SCHISM	38	63%	59	79%	64	100%	58	94%	65	97%	71	91%
PROCLUS	60	100%	62	83%	46	72%	62	100%	47	70%	77	99%
P3C	39	65%	45	60%	36	56%	58	94%	63	94%	77	99%
STATPC	31	52%	62	83%	57	89%	58	94%	41	61%	78	100%

Figure 2.7: Clustering quality (F1 and accuracy) on real world data. Captions: data set (size; dimensionality)

INSCY and FIRES cannot compete with RESCU. The high quality of RESCU is not only a result of the density-based instantiation of the subspace cluster definition but in particular due to our new relevance model.

In Figure 2.7 we show F1 and accuracy results for six further real world data sets. In addition to the absolute values we note the relative quality compared to the best measurement on each data set. Best 95% results are highlighted in gray. RESCU achieves top quality results for *all* data sets with respect to *both* measures. Competing approaches show highly varying performance. None of them achieves top quality allover. Although some of the approaches achieve slightly better results on some of the data sets, RESCU reliably shows top results on all data sets.

2.4.3 Parametrization

We also evaluate the approximation quality of RESCU compared to the full optimization model. As we have proven RESCU is NP-hard and thus we can compute an optimal solution only for very small settings. In this experiment we use datasets with 10 clusters. The cost and objects per cluster are randomly selected from $(0; 1]$ and $[1; 100]$, respectively. For each Δ value we generate at least 20 random datasets

to calculate the optimal and approximate solution. Figure 2.8 gives the relative approximation quality compared to the optimal solution (in terms of cost) for different values of Δ. Note that we have to solve a minimization problem, i.e. the optimal solution corresponds to the lower curve (100%). On average, the approximative solution shows only small differences to the optimal solution. The whiskers, corresponding to the 15% and 85% quantiles respectively, indicate that also rare cases yield good approximation qualities. The approximation quality is remarkably robust against the parameter Δ, the cluster gain threshold. Even for extremely rigid values close to zero, average approximation is close to optimal.

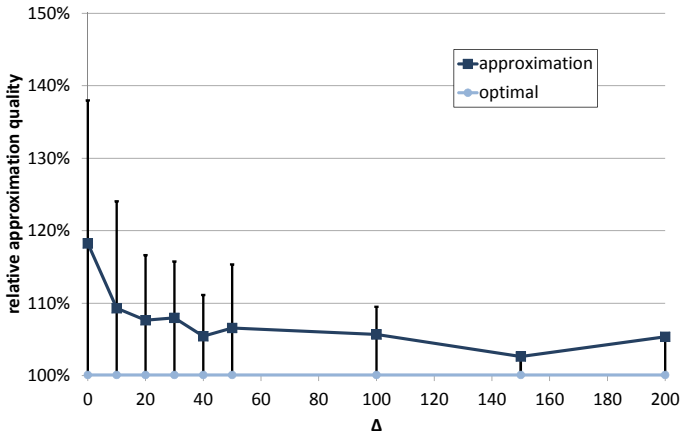


Figure 2.8: Approximation quality (Δ variation)

Next, we evaluate the flexibility of our RESCU model. For our cost function instantiation (cf. Sec. 2.2.6), we vary the β parameter that controls interestingness as a trade-off between higher dimensional clusters and more objects per cluster. For the 32d pendigits data set, Figure 2.9 shows the median size (triangles) of the clusters and the number of high-D (≥ 12) clusters (squares) for the relevant clustering of a given β value. As we can see, low β values give strong preference to large and few clusters, whereas high values result in many clusters with less objects. Likewise, RESCU can be just as easily adapted using any other cost function that reflect the interestingness in a user's analysis.

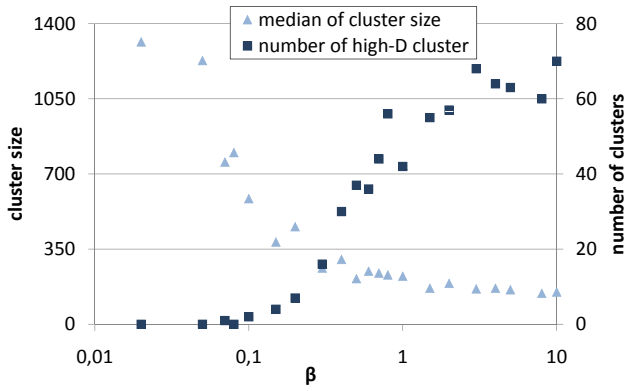


Figure 2.9: Effects on clustering by β variation

2.5 Conclusion

In this chapter we introduced the *RESCU* (relevant subspace clustering) model for mining the most interesting non-redundant clusters in high dimensional data. Our model incorporates both non-redundancy and interestingness via a new cluster gain definition. We prove that the computation of this model is NP-hard. For *RESCU*, we propose an approximative solution that shows high accuracy with respect to our relevance model. Thorough experiments demonstrate that *RESCU* reliably outperforms existing subspace and projected clustering algorithms while automatically reducing the output to all and only relevant clusters.

Overall, our RESCU model solves the challenge of redundancy in the clustering result but avoids the limitations of projected clustering by allowing overlapping clusters. Though, inspired by the Set Cover idea, our model is not able to detect (all) multiple views in the data, i.e. some clusters might be lost due to our strict redundancy handling. Thus, in the following chapters we introduce novel models to handle redundancy in subspace clustering results more appropriate and to enable the detection of multiple views.

Chapter 3

Detection of Orthogonal Concepts

In today's applications, data is collected for multiple analysis tasks. Thus, for each object one gathers many measurements in one high dimensional database to provide a large variety of information. In such scenarios one typically observes that each object can participate in various groupings, i.e. objects fit in different roles. For example, in customer segmentation, we observe for each customer multiple possible behaviors which should be detected as clusters. Each of these multiple clustering solutions is valuable and interesting as different perspectives on the same data are given. Especially for high dimensional data, where each object is described by multiple attributes, alternative clusters in different attribute subsets are of major interest. Recent clustering techniques, however, do not succeed in detection of these orthogonal concepts hidden in the data. They either miss multiple concepts for each object, provide redundant clusters in very similar subspaces, or transform the data thus losing the original descriptive attributes.

In this chapter, we propose a novel clustering method aiming only at orthogonal concept detection in subspaces of the data. Unlike existing clustering approaches, OSCLU (Orthogonal Subspace CLUstering) detects for each object the (almost) orthogonal concepts described by differing attributes while it prunes similar concepts. Thus, our clustering method is able to detect clusters hidden in multiple views of the data but simultaneously avoids redundant information. Thorough experiments on real and synthetic data show that OSCLU yields substantial quality improvements over existing clustering approaches.

3.1 Motivation

Traditional clustering approaches compute a partitioning of the data, grouping each object in at most one cluster or detecting it as noise. However, it is not always the case that an object is part of only one cluster. Multiple meaningful groupings might exist for each object. The detection of such multiple clusters describing different views on each object is still an open challenge in recent applications.

Since data is often collected for multiple analysis tasks, databases contain objects specified by very many attributes. As one does not know the hidden structure of the data, one mixes up different measurements in one high dimensional data set. Thus, each object can participate in various groupings reflected in different subsets of the attributes. For example, in customer segmentation, objects are customers described by multiple attributes specifying their profile. A customer might be grouped by the attributes "average fruit consumption" and "sport activity" with other "healthy people" having high values in both of these attributes. The same customer might be a "Rock Fan" which could be specified by high values in the attribute "attendance to rock concerts" and low values in "attendance to classic concerts" (cf. Fig. 3.1). We observe for each customer multiple possible behaviors which should be detected as clusters. Thus, clusters may overlap in their clustered objects, i.e. each object may be represented in multiple clusters. Furthermore, each behavior of a customer is described by specific attributes. Thus, meaningful clusters appear only in these specific subspace projections of the data. While the attribute "attendance to rock concerts" is useful for the distinction of musical interests, the attribute "fruit consumption" is irrelevant for grouping musical interests of customers.

Detection of Orthogonal Concepts

We generalize these observations as they are not only applicable to customer segmentation. In other applications, objects might be sensor nodes represented by multiple sensor measurements, or objects might be genes described by their expression level under multiple conditions. For each of these application scenarios, objects are described by very many attributes. For such high dimensional, clusters cannot be identified in the full dimensional space but each of the objects might be part of different groups in different subsets of attributes. In general, we call this an object that is part of multiple *orthogonal concepts* that provide different *views* on the data. For the general case of high dimensional data, a concept is characterized by a subset of attributes and by the subspace clusters located in this (or a similar)

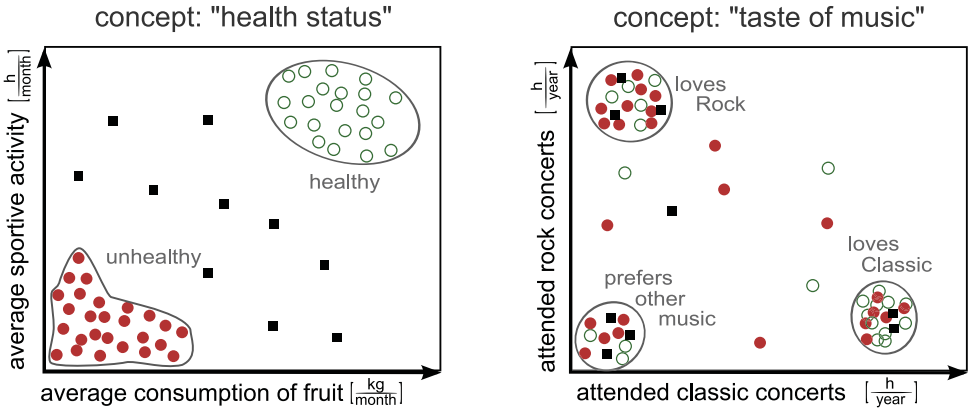


Figure 3.1: Different concepts providing different views on the data

subspace. The concept's dimensions provide the reasoning why specific objects are grouped together. In Fig. 3.1 the concept "taste of music" is described by two dimensions. Overall, we can substitute the detection of orthogonal concepts with the detection of *orthogonal subspaces* and their contained clusters.

A concept can contain several clusters that are clearly separated in the relevant dimensions of the concept, like customers loving Rock or customers loving Classic in our previous example. Each object may be clustered by at most one cluster in the same subset of relevant attributes. However, it can be clustered in multiple orthogonal concepts having different sets of attributes. Considering the concept "health status", a "Rock Fan" can be clustered with other customers to form a new grouping of "healthy people". There might exist multiple meaningful groups for each object as it can be interpreted in multiple different ways. The same observation can be made in the other scenarios as well: Genes are controlling multiple functions (concepts) expressed only under specific conditions (relevant attributes for the concept), or sensors are measuring multiple concurrent environmental events (concepts) specified by different sensor measurements (relevant attributes).

Overall, orthogonal concepts provide the user different views on the same data by determining for each object different groupings using no or only few shared attributes. Our OSCLU (Orthogonal Subspace CLUstering) approach is able to detect such multiple orthogonal concepts. Each detected cluster provides novel information, as we select only clusters in (almost) orthogonal subspaces and prune redundant clusters in similar subspace projections of already detected clusters.

Summing up, in our approach we aim at detection of only the orthogonal concepts fulfilling the following properties:

- subspaces and subspace clusters represent the concepts in the database
- objects might be present in multiple clusters, if the subspaces of their concepts differ (to a high extent)
- each cluster provides novel information for its concept

Following these properties, we propose a method for selection of orthogonal subspaces by using a similarity measure on subspace projections. Our novel approach OSCLU chooses according to this similarity only the orthogonal subspaces for the result set. In addition, we propose a relaxation of the orthogonal subspaces to "almost orthogonal subspaces". This generalization allows us to detect concepts sharing a certain amount of common dimensions. The attribute "gender" for example could belong to several concepts. Relaxing to almost orthogonal subspaces includes more possible concepts in the result.

As each object might be present in multiple clusters, we have to ensure that each cluster adds sufficiently novel information within its concept. Unlike most subspace clustering techniques we prevent redundant information. For this purpose we introduce an interestingness measure for choosing only sufficiently distinguishing clusters from similar concepts. Furthermore, to select the most interesting clusters, we present an objective function that is based on multiple properties like size, dimensionality and density extracted out of the subspace clusters.
Using both properties of orthogonal subspaces and most interesting clusters, OSCLU performs a global optimization of the result set. It ensures to include overlapping clusters to detect multiple views on the data. Furthermore, it prunes similar subspaces and non-interesting clusters to ensure only the meaningful patterns in the result.

3.2 Comparison with Related Work

Different clustering paradigms have been proposed in the past decades. In this section, we review the main techniques and show their limitations in detection of orthogonal concepts in high dimensional data. Especially, we show the differences to our approach as none of the proposed techniques analyzes subspace projections to steer cluster detection in the direction of orthogonal concepts.

Traditional Clustering. Traditional clustering approaches, aim at the detection of clustered objects using all attributes in the full data space. Several dif-

ferent clustering models, like *partitioning* or *density-based* clustering, have been proposed. However, independent of the underlying clustering model, full space clustering approaches do not scale to high dimensional data spaces covering multiple different concepts since clusters do not appear across all attributes.

Clustering in Subspace Projections. Recent research for clustering in high dimensional data has introduced a number of different approaches summarized in [PHL04, KKZ09] and they were already discussed in Chapter 2.1. Their common goal is to detect clusters in any (axis-aligned) subspace projection of the data. Each cluster is associated with a set of relevant dimensions in which this pattern has been discovered. The existing approaches fail in detecting the orthogonal concepts in the data: Either they are only able to detect (almost) disjoint clusters and hence are unable to detect multiple clusters per object, or they allow objects to be part of multiple clusters but the result contains a huge number of redundant clusters. Although some recent approaches as our RESCU model or [AKMS07, AKMS08a, MS08] confine the result to only non-redundant clusters the similarity of the sets of dimensions is not adequately taken into account. Thus, a set of objects might result in being part of clusters with very similar subspaces. Taking similarity of subspaces into account, is essential for effective detection of only the orthogonal subspaces.

Multi-view Clustering. There exist some recent extensions of traditional clustering techniques that try to detect multiple clustering solutions. They can be divided into two categories: methods operating in the original full-space [BB06, DB10b, DB10a, NE10] and methods using space transformations [DQ08, QD09, CFD07]. Approaches from the first category are problematic since they analyze just a single subspace to detect multiple clustering solutions of high quality. In a single subspace, however, the resulting groupings are either very similar or the quality of the alternative solutions is very low. Furthermore, as discussed in Chapter 1, clustering in the full-space is questionable due to the curse of dimensionality [BGRS99].

Approaches from the second category try to iteratively detect further clustering solutions by transforming the data space based on the previously detected clustering. Thus, they force the traditional clustering algorithm to find novel clusters in a novel space. In contrast to subspace clustering algorithms, they search for clusters in the full space or in arbitrary space transformations. By transforming the data they pose constraints on the cluster detection. However, transformation hinders the interpretation of results as the original attributes are not directly used

for description of detected concepts. Furthermore, due to the iterative procedure, these approaches typically detect some of the already detected clusters multiple times [CFD07] or they are restricted to one alternative clustering for a given set of clusters [QD09]. Detection of only orthogonal concepts is hindered as redundant information does not provide any additional knowledge and restriction to a fixed number of alternative concepts does not provide knowledge about all hidden concepts. Overall, simultaneously detecting all orthogonal concepts without including redundant information is not possible.

3.3 Orthogonal Concepts in Subspaces

In this section, we present our model for the detection of orthogonal concepts in subspaces of high dimensional data. Formally we map our contributions to an optimization problem based on detected subspace clusters in the database. In contrast to subspace clustering, where all clusters are selected for the result set, we choose only a subset of most interesting clusters based on orthogonal subspaces. For this purpose we make a distinction between the cluster definition and clustering definition. While the cluster model defines the properties that a set of objects $O \subseteq DB$ and a set of dimensions $S \subseteq Dim$ have to fulfill to be a valid cluster $C = (O, S)$, the clustering model determines a set of clusters $M = \{C_1, \ldots, C_n\}$ to be a valid clustering result. The valid clustering result for traditional subspace clustering is simply the set All that contains all subspace clusters. This set is highly redundant and hence in our model it is not a valid clustering.

We want to generate an informative clustering $Opt \subseteq All$ so that the clusters in the result set represent the multiple concepts of the data without obfuscating this structure by redundant information. As motivated before, each object might be present in multiple clusters if the clusters describe different concepts, and each cluster $C \in Opt$ has to provide novel information within its concept. In short: it is not allowed to group the same objects in similar subspaces by several clusters. Therefore, we have to define

- whether a concept is similar to another one or if it describes a different concept, and
- if a subspace cluster identifies a new grouping within its concept.

As a consequence, overlapping clusters between different concepts are possi-

ble, in contrast to projected clustering. We solely have to check if the same objects are already described within similar concepts to filter out uninteresting clusters and to steer our cluster detection to the orthogonal subspaces. Thus, in a first step in Section 3.3.1, we define the notion of (almost) orthogonal concepts to determine which concepts are similar to a selected one. In Section 3.3.2 we present the interestingness criterion, that each cluster has to fulfill to be an informative cluster within its concept. In Section 3.3.3 we define our overall model for the optimal orthogonal clustering and we prove in Section 3.3.4 that solving this model is NP-hard. Finally, we discuss the parameters of our approach in Section 3.3.5 and show in Section 3.3.6 how the user can influence the clustering result.

3.3.1 Almost orthogonal concepts

The data collected in today's applications are generated by different concepts which are mixed together. In an optimal setting, the concepts, described by subspaces, share no dimensions; and we can clearly distinguish between them. If we identify a concept in the subspace S, all other subspaces T that share at least some dimensions $T \cap S \neq \emptyset$ are similar to it and we can prune them. T cannot characterize a different concept because a dimension $d \in S \cap T$ is already covered by the concept in S and, hence, T does not detect a novel concept in this scenario. Hence, all subspaces that are similar to S are excluded from further consideration by the identification of S. This can be formalized by:

$$\begin{aligned} coveredSubspaces_0(S) &= \{T \subseteq Dim \mid T \cap S \neq \emptyset\} \\ &= \{T \subseteq Dim \mid |T \cap S| > 0\} \end{aligned} \tag{3.1}$$

A concept with the relevant subspace T is orthogonal to a concept in S if $T \notin coveredSubspaces_0(S)$. The dimensions of T and S are disjoint and hence we can detect novel information in T. So our clustering model only has to identify clusters in subspaces which are orthogonal and prune the already covered subspaces.

However, this orthogonality definition is a too hard restriction for our clustering model. Many subspaces are prohibited for selection and hence the resulting clustering contains only low information. By definition each dimension appears in at most one concept. However, overlapping concepts are useful and expected in real life scenarios, e.g. the attribute "gender" in a customer database could appear in multiple concepts. For subspace clustering we need a relaxation of the

orthogonality property.

A less hard restriction is realized by the idea of excluding lower dimensional projections of S. The subspace S is more meaningful for the representation of a concept than its projections, which contain fewer attributes. Hence, if we identify S as the relevant subspace for a concept, each projection is already described by this subspace. The subspaces similar to S can be defined by:

$$\begin{aligned} coveredSubspaces_1(S) &= \{T \subseteq Dim \mid T \subseteq S\} \\ &= \{T \subseteq Dim \mid T \cap S = T\} \\ &= \{T \subseteq Dim \mid |T \cap S| = |T|\} \end{aligned} \tag{3.2}$$

By this definition we can find overlapping concepts, e.g. characterized by $S_1 = \{1, 2\}$ and $S_2 = \{2, 3\}$. Neither of them is similar to the other concept and, hence, both of them could appear in the result set. This definition is related to the maximality property in other subspace clustering approaches [AKMS07, AKMS08a], resulting in the same problems. Even if two subspaces share a high fraction of dimensions, e.g. 9 out of 10, they represent different concepts. Thus, similarity of subspaces is not yet modeled in an adequate way.

Our model of almost orthogonal concepts integrates the advantages of both models. We allow overlapping concepts, but we also avoid concepts with too many shared dimensions. Thus, we only include (almost) orthogonal concepts in the result and obtain a flexible model by generalizing both definitions to:

$$coveredSubspaces_\beta(S) = \{T \subseteq Dim \mid |T \cap S| \geq \beta \cdot |T|\}$$

with $0 \leq \beta \leq 1$. For $\beta \to 0$ we get Equation 3.1, for $\beta = 1$ the Equation 3.2. The case $\beta = 0$ is possible but rarely useful: In this case, all concepts are similar to each other even if they share no dimensions. Thus, the final result can just contain a single concept, similar to the RESCU approach. Since we focus on the detection of multiple concepts, we will not consider the case $\beta = 0$ in the following.

The idea of our clustering model is to avoid the grouping of the same objects in similar concepts by several clusters. Given a cluster C we have to determine the set of clusters that are in similar concepts. Because we use (almost) orthogonal subspaces for the orthogonal concept detection, we can determine these clusters by checking if their subspaces cover the subspace of C. We call this set the concept group of C which can be formalized by the following definition.

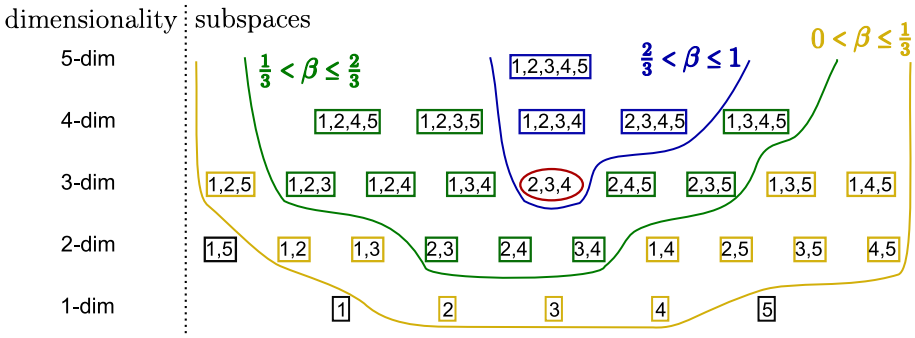


Figure 3.2: Concept group with variation of β

Definition 3.1 *Concept group*
The concept group of $C = (O, S)$ *with respect to a set of clusters* $M = \{C_1, \ldots, C_n\}$ *is defined as*

$$conceptGroup(C, M) = \{C_i \in M \backslash \{C\} \mid S \in coveredSubspaces_\beta(S_i)\}$$

The concept group of $C = (O, S)$ contains all clusters that share at least a β-fraction of the dimensions of S. Checking the grouped objects O of C against the objects of clusters in C's concept group is required to provide novel information within similar concepts. All other clusters, not in the concept group of C, do not need to be considered because they belong to other concepts. Thus, we permit multiple concepts in our result.

Let us consider Figure 3.2 where the selected cluster C is in the subspace $\{2, 3, 4\}$. For $\beta \to 0$ we have to compare C with all clusters in subspaces sharing at least one dimension. C has to group new objects w.r.t. these clusters because they all characterize similar concepts. The higher β the less subspaces are considered as similar and hence the more concepts may occur in the final clustering. The choice of $\beta = 1$ results in comparing C only to higher dimensional clusters C', which cover the subspace of C by their dimensions. For example, the concept described by the subspace $\{1, 2, 3, 4\}$ subsumes the concept of C and, thus, C has to be checked against this subspace. Thereby, we see that the notion of concept group is not symmetric, i.e. $C' \in conceptGroup(C, M)$ but $C \notin conceptGroup(C', M)$ might hold. The concept group of a cluster C tends to include more higher dimensional clusters. The concept group of a low dimensional cluster, that is in general less interesting, usually contains more clusters compared to the one of a higher dimen-

sional cluster. Thus, for a low dimensional cluster it is more difficult to provide novel information and consequently to be included in the result set.

3.3.2 Global interestingness

After defining the clusters $\mathcal{D}$ that are located in similar concepts as C, we have to ensure that the cluster C is interesting enough compared to the clusters in $\mathcal{D}$. For our resulting clustering $Opt \subseteq All$, each cluster $C \in Opt$ has to fulfill this property since otherwise redundant information is included. According to our motivation, a cluster $C = (O, S)$ has to group new objects compared to the clusters located in similar concepts. Hence we use the coverage of objects as a criterion for interestingness. For a clustering $M = \{C_1, \ldots, C_n\}$ the coverage is defined as:

$$Cov(M) = \bigcup_{i=1}^{n} O_i$$

Because a strict partitioning of the clusters in similar concepts is not useful, i.e. we would enforce that each object of C is in no other cluster, we relax this property. We calculate the relative fraction of objects which are not covered by other clusters in similar concepts w.r.t. the whole cluster size.

Definition 3.2 *Global interestingness*
Given a cluster $C = (O, S)$ and a set of clusters $M = \{C_1, \ldots, C_n\}$. The global interestingness of C with respect to M is

$$I_{global}(C, M) = \frac{|O \backslash Cov(conceptGroup(C, M))|}{|O|}$$

First, we determine the clusters in similar concepts to the one of C, and afterwards, their objects are removed from O to obtain the newly covered objects of C. Only if $I_{global}(C, M)$ is larger than a given threshold α the cluster adds sufficiently new information to this concept.

Figure 3.3 illustrates this interestingness check. Let us assume that M contains the clusters C_7 to C_{10} and possible further clusters in other subspaces (not within dimension 3). If we choose $C = C_{10}$, the concept group corresponds to $\{C_7, C_8, C_9\}$. The remaining clusters are not considered because they represent other concepts. C_{10} has to group new objects within the concept. However,

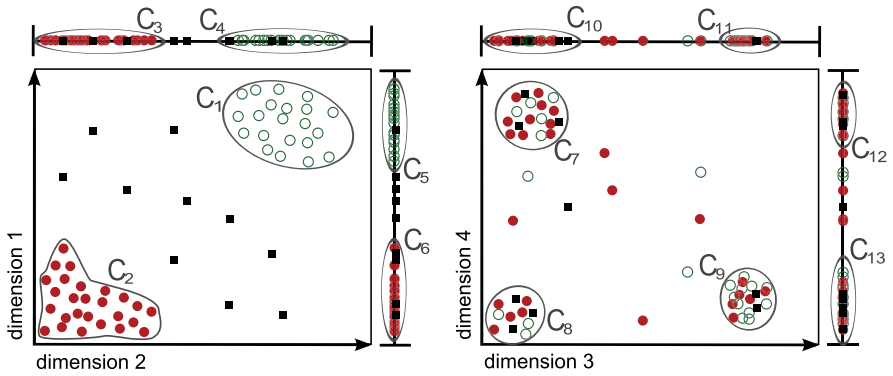


Figure 3.3: Example for the selection of an optimal orthogonal clustering

most of the objects (29 out of 32) from C_{10} are already covered by the other clusters and, hence, the information obtained by C_{10} in this concept is small ($I_{global}(C, M) = \frac{32-29}{32}$). For a threshold $\alpha > \frac{3}{32}$, the cluster C_{10} is regarded as redundant with respect to M.

The user is able to control the required interestingness of a cluster by variation of α. If the fraction of newly clustered objects is smaller than α we do not choose the cluster. For the extremal value $\alpha = 1$, the clusters in similar concepts must not overlap. For $\alpha \to 0$ a cluster is selected as long as not all objects are covered by other clusters. Consequently a high overlap is possible.

An important aspect of this model is that the interestingness is simultaneously checked against several clusters. Unlike other models [AKMS07, AKMS08a] that make only a pairwise comparison of the object coverage, in our model all clusters from a similar concept are considered at the same time to evaluate the interestingness of the new cluster. If we did not check against several clusters, the cluster C_{10} in Figure 3.3 would get a misleading high interestingness value. A pairwise comparison of C_{10} to C_7 or C_8 indicates a high fraction of newly clustered objects which is in fact not true. In contrast, the calculation of our global interestingness value is in line with the observations made for the global redundancy removal as introduced in our RESCU approach (cf. Chapter 2).

Let us choose a clustering $M \subseteq All$. The global interestingness ensures that each cluster $C \in M$ results in an information gain within its concept by covering new objects. Thus, we do not include redundant information. At the same time, however, varying concepts are possible in M and considered by the definition. Thus, the proposed properties for a good clustering (cf. page 50) are realized.

Definition 3.3 *Orthogonal clustering*
The clustering $M = \{C_1, \ldots, C_n\}$ is orthogonal iff

$$\forall C \in M : I_{global}(C, M \backslash \{C\}) \geq \alpha$$

The clustering $M = \{C_1, C_2, C_7, C_8, C_9\}$ in our example from Fig. 3.3 is an orthogonal clustering, while the clustering $M \cup \{C_{10}\}$ is not. However, the proposed definition alone is not yet sufficient to determine an optimal clustering $Opt \subseteq All$. Several clusterings could fulfill the definition, e.g. the trivial clustering $M = \emptyset$. The user wants to get an overview of the clustering structure and seeks for the most informative clusters. We have to ensure that these clusters are selected.

3.3.3 Optimal orthogonal clustering

While the global interestingness $I_{global}(C, M)$ always rates the cluster C with respect to a clustering M, we now assess the interestingness of the cluster C on its own. This so called local interestingness should correspond to the user-specific notion of interesting clusters. Formally, we define a function I_{local} which maps each cluster C to the value $I_{local}(C)$. This function could include different aspects, as the dimensionality or the size of the clusters. A discussion of this function is presented in Section 3.3.6.

Both, the global and local interestingness are used to define our optimal orthogonal clustering. With the global property we ensure that only informative clusters within similar concepts are selected. At the same time we want to maximize the sum of the local interestingness for the resulting clusters. By maximizing the local interestingness we get the most interesting clusters but also as many interesting clusters as possible (taking the orthogonal clustering constraint into account).

Definition 3.4 *Optimal orthogonal clustering (OOC)*
Given the set All of all possible subspace clusters, a clustering $Opt \subseteq All$ is an optimal orthogonal clustering iff

$$Opt = \arg \max_{M \in Ortho} \left\{ \sum_{C \in M} I_{local}(C) \right\}$$

with $Ortho = \{M \subseteq All \mid M$ is an orthogonal clustering$\}$

In Figure 3.3 we show an overall example with $\alpha = 0.5$ and $\beta = 0.5$. The clustering $M_1 = \{C_1, C_2, C_7, C_8, C_9\}$ is a valid orthogonal clustering, because each cluster covers a sufficient amount of new objects within its concept. Although C_1 and C_9 contain similar objects the overlap is permitted because different concepts are realized. The clustering $M_1 \cup \{C_{10}\}$ for example is not valid. As shown in our previous example it holds $I_{global}(C_{10}, M_1) = \frac{32-29}{32} < \alpha$. Obviously each subset of M_1 is also an orthogonal clustering but less informative than M_1. Hence, these subsets cannot be optimal clusterings. If we assume that the user is more interested in high dimensional clusters and chooses I_{local} accordingly, the sum $\sum_{C \in M_1} I_{local}(C)$ would be maximal out of all orthogonal clusterings. Another orthogonal clustering like $M_2 = \{C_1, C_2, C_{10}, \ldots, C_{13}\}$ which contains the one-dimensional projections from the second concept would therefore result in a lower value for the sum. As a consequence M_1 is preferred over M_2, and M_1 is the optimal clustering in this example.

Our model provides a selection of only interesting clusters in different and novel concepts. An overwhelming result size of redundant clusters is prevented. As we use subspace clusters in our model, the interpretability of the result set and the identification of the relevant attributes for each concept are guaranteed. Unlike other orthogonal clustering models we keep the original dimensions and we use them for the orthogonality check. We steer the cluster selection to orthogonal subspaces.

3.3.4 Complexity analysis

In this section, we prove the NP-hardness of our optimal orthogonal clustering problem (denoted as OOC). To achieve this aim, we reduce the NP-complete $SetPacking$ problem [GJ79] to our model, i.e. we show $SetPacking \leq_P OOC$. Given several finite sets O_i the $SetPacking$ problem seeks for the maximal number of disjoint sets.

Theorem 3.1 *Computing OOC (Def. 3.4) is NP-hard.*

Proof 3.1
We show that $SetPacking \leq_p OOC$.

A. Input mapping: *Each set O_i is mapped to the cluster $C_i = (O_i, \{1\})$. Furthermore we set $\beta \in [0, \ldots, 1]$, $\alpha = 1$ and $I_{local}(C) = |S|$.*

B. OOC ***generates a valid*** $SetPacking$ ***solution:***

1) The concept group contains all clusters:

$conceptGroup(C, M\backslash\{C\})$
$= \{C_i \in M\backslash\{C\} \mid S \in coveredSubspaces_\beta(S_i)\}$
$= \{C_i \in M\backslash\{C\} \mid \{1\} \in coveredSubspaces_\beta(\{1\})\}$
$= M\backslash\{C\}$

2) Each orth. clustering M *contains only disjoint sets:*

M *is orthogonal clustering*
$\Leftrightarrow \forall C \in M : I_{global}(C, M\backslash\{C\}) \geq 1$
$\Leftrightarrow \forall C \in M : \frac{|O\backslash Cov(conceptGroup(C, M\backslash\{C\}))|}{|O|} \geq 1$
$\Leftrightarrow \forall C \in M : |O\backslash Cov(M\backslash\{C\})| \geq |O|$
$\Leftrightarrow \forall C \in M : O_\cap \bigcup_{C_i \in M\backslash\{C\}} O_i = \emptyset$

3) Opt *contains maximal number of such disjoint sets:*

$Opt = \arg\max_{M \in Ortho}\{\sum_{C \in M} I_{local}(C)\}$
$\Leftrightarrow Opt = \arg\max_{M \in Ortho}\{\sum_{C \in M} |\{1\}|\}$
$\Leftrightarrow Opt = \arg\max_{M \in Ortho}\{\sum_{C \in M} 1\}$
$\Leftrightarrow Opt = \arg\max_{M \in Ortho}\{|M|\}$

(2) and (3) $\Rightarrow$ Opt *is a valid* $SetPacking$ *solution* $\Rightarrow$ OOC *is NP-hard* □

3.3.5 Discussion of parameters

To control the optimal orthogonal clustering result, the user can vary two parameters: α and β. The parameter α intuitively controls the allowed overlap between clusters that are located in similar subspaces. The fraction of newly covered objects within each cluster has to be at least α. Increasing the value of α leads to a stricter evaluation of redundancy. Thus, for $\alpha = 1$ the result must only contain disjoint clusters (w.r.t. the corresponding concept) while $\alpha \to 0$ permits almost arbitrary overlapping clusters. Analogously, β controls the overlap between the concepts of the final clustering solution. For $\beta \to 0$ the concepts have to be disjoint to be regarded as different. By increasing the value of β, the final clustering is allowed to contain concepts that share a certain amount of dimensions. Intuitively, α controls the redundancy between clusters while β controls the redundancy between concepts.

As a guideline to select these parameters in real world applications, we recommend the selection of α close to 1 and β close to 0. The obtained clustering

solution is easy to interpret since it intuitively corresponds to the following procedure: First, partition the dimensions of the database into different views. Second, perform projected clustering on each of these views. Our method, however, is superior since the number of views and their subspaces are automatically determined as well as noisy dimensions are excluded.

3.3.6 Instantiation of our model

Before we present our local interestingness function, we first set up our cluster definition. We use the same density-based clustering model as introduced in Chapter 2, Definition 2.8. Thus, the density of an object o in a subspace S is the number of objects in its adaptive ε-neighborhood around it, i.e. $density^S(o) = |\{p \in DB \mid dist_S(o,p) \leq \varepsilon(|S|)\}|$.

With this cluster definition we can define our user-specific local interestingness function. Three main properties characterize a subspace cluster $C = (O,S)$ in our cluster instantiation. The dimensionality $|S|$, the size $|O|$ and the density. A very dense cluster shows small variation in the attribute values of the relevant dimensions and hence is more interesting than a sparse cluster. We use the mean density $\frac{1}{|O|}\sum_{o\in O} density^S(o)$ over all objects within the cluster for this criterion.

Maximizing all measures at the same time is in general not possible, e.g. low dimensional clusters are usually larger than high dimensional clusters. Therefore, our local interestingness function subsumes all measures and gives the user the flexibility to weight the measures dependent on the application. The local interestingness function used in our experiments is

$$I_{local}(C) = |S|^a \cdot |O|^b \cdot \left(\frac{1}{|O|}\sum_{o\in O} density^S(o)\right)^c$$

with $C = (O,S)$ and $a + b + c = 1$.

3.4 The OSCLU Algorithm

The optimal orthogonal clustering has global properties that increases the computational complexity. As we have already proven, the problem is NP-hard, and hence, we cannot expect that an efficient algorithm exists. Furthermore, we cannot generate the huge set of all subspace clusters All in a first step and select

the optimal subset afterwards. We develop a heuristic algorithm (OSCLU) that incrementally adds further clusters to the result set. For efficient calculation we integrate the clustering process into the process of concept and cluster selection. This means that not all clusters in all subspaces are generated but many subspaces are pruned based on already detected concepts/clusters. An important question is which subspaces should be clustered first and, hence, which clusters should be added at the beginning to the result set in order to prune many other subspaces.

Traditional bottom-up approaches that start with the low dimensional clusters are not useful for pruning based on our global interestingness criterion. As already mentioned, the concept group of a cluster contains mainly higher dimensional clusters (cf. Fig. 3.2). Thus, a low dimensional cluster has to compare its object coverage against more clusters than a high dimensional cluster. A low dimensional cluster is more likely to be excluded from the result set than a high dimensional cluster. For this reason we use a top-down approach to add clusters to the final clustering.

Our algorithm, summarized in Algorithm 3.1, comprises three major contributions to avoid clustering of all subspaces. First, we develop a ranking of the subspaces (all with the same dimensionality) without clustering them (lines 4-6). The ranking accounts for the similarity of the current subspace with already detected concepts. The greater the number of already detected similar concepts, the less interesting is the subspace for a further investigation. In a second step, the ranking considers the possibility for a good clustering in a subspace based on an efficient estimation. After ranking the subspaces in a priority queue we use the first subspace for clustering (line 8). If clusters were identified, we incrementally update the result set (line 10). We have to consider the global interestingness so that redundant clusters are not selected. Furthermore, a high local interestingness of the selected clusters should be ensured. Resorting the ranking and the possible pruning of further subspaces (line 11) based on the new clusters is performed in order to push novel concepts to the top of the priority queue. If all subspaces with the dimensionality dim are pruned or selected for clustering, we decrease the dimensionality to realize the top-down approach.

As an additional step we present an efficient method that approximately identifies the highest dimensionality (of a subspace) in which clusters are expected (line 2). This avoids to start our ranking in the full-space, where clusters are in general not present.

Algorithm 3.1: OSCLU (Orthogonal Subspace CLUstering)

```
result set M := ∅
find initial dimensionality dim                    (Sec. 3.4.3)
while dim > 0 do
    rank and prune subspaces based on              (Sec. 3.4.1)
        1) subspace orthogonality score
        2) subspace quality score
    while ranking not empty do
        choose best subspace for clustering
        if clusters found then
            update result set M                    (Sec. 3.4.2)
            resort ranking and prune
    dim=dim-1
return result set M
```

3.4.1 Orthogonal Subspace Selection

Clustering each subspace is not efficient since many subspaces can be pruned because of similar concepts that are already detected. We use two techniques to rank subspaces without clustering. The aim is to cluster only interesting and orthogonal subspaces. In our first approach we use the similarity of already discovered concepts for pruning and ranking. The larger the number of similar subspaces in the result set the higher is the possibility that new clusters in the current subspace cover the same concept and hence provide no novel information. We define the orthogonality score of a subspace S w.r.t. the current result set M as

$$orthogonality_score(S, M) = \\ |\{T \subseteq Dim \mid S \in coveredSubspaces_{\beta}(T) \wedge \exists (O, T) \in M\}|$$

The definition is similar to the concept group, but only considers the subspaces instead of the actual subspace clusters. The higher the score the worse is a subspace because many similar concepts are already in the result set. If the score is too high, a subspace is pruned. The orthogonality score is the first criterion for ranking.

During a run of the algorithm the result set M changes and hence the orthogonality score does so, too. This way, only subspaces representing novel concepts are top ranked and potentially clustered. The clustering is concentrated to the

orthogonal and novel concepts.

Our second approach makes use of subspace search [CFZ99, KKKW03] for measuring the quality of subspaces. Usually subspace search is a stand-alone technique for identifying interesting subspaces. Each subspace is mapped to a quality value, a high value corresponds to a high possibility for a good clustering structure. Our trick is to use this technique within the clustering task. We guide our algorithm to cluster only the most interesting subspaces based on the calculated qualities. Therefore, our ranking is extended such that all subspaces with the same orthogonality score are secondly ranked based on this qualities. In total, our ranking concentrates not only on novel concepts but also on high quality subspaces.

The subspace search method within our framework is easily exchangeable and we can use techniques like RIS [KKKW03] or ENCLUS [CFZ99]. Similar to the method of [KKKW03], we exploit our density-based cluster definition (cf. Sec. 3.3.6) to evaluate the quality of a subspace. To find clusters in a subspace, several objects must have a high density according to our density-based clustering model, i.e. for an object p the value of $density^S(p)$ is large. We use a strategy that randomly selects points and calculates their mean density. This method is efficient and represents a good indicator for the existence of clusters. The higher this quality score the better the subspace. Let $Seeds$ be a set of randomly selected points, the quality score is then defined as

$$quality_score(S, M) = \frac{1}{|Seeds|} \sum_{p \in Seeds} density^S(p)$$

In contrast to the method of [KKKW03], which just considers the number of dense objects in a subspace, we compute the average over the exact density values.

3.4.2 Incremental result construction

After ranking the subspaces based on the two scores in a priority queue, we select the top-ranked subspace and cluster it according to our model. We get a list of resulting clusters New. We now have to check which clusters $C \in New$ should be included to our result set M. In a first step we analyze the global interestingness of the new clusters. For each cluster $C \in New$ we calculate $I_{global}(C, M)$, and we distinguish two cases.

If $I_{global}(C, M) \geq \alpha$ we directly add the cluster to M, i.e. the new result set is $M := M \cup \{C\}$. The cluster C adds sufficiently new information. This way, we

ensure that in each step of the algorithm only informative clusters are selected. Please note that by this procedure the Definition 3.3 is relaxed. We do not check the global interestingness of the remaining clusters in M which could be changed by selection of C. This recalculation would be too costly. However, due to our top-down approach, higher dimensional clusters are added first to M, and these clusters are rarely removed by low dimensional clusters. Additionally, within the same dimensionalities our ranking tries to rank the best subspaces on top and, hence, these clusters are selected first from the priority queue.

If $I_{global}(C, M) < \alpha$ we do not reject the cluster immediately but we perform an additional improvement heuristic. We want to maximize the local interestingness in our model, hence we check if it is possible to remove some clusters from M such that C is afterwards globally interesting and the sum of the local interestingness is increased. The pseudo-code of our algorithm that decides if C is included and which subset of M should be removed is depicted in Algorithm 3.2. First, we rank the clusters from $conceptGroup(C, M)$ in decreasing order based on their local interestingness values. Second, we select the most interesting clusters which do not result in redundancy for C (set N). The clusters which induce the redundancy are stored in R. At the end it holds that $I_{global}(C, M \backslash R) \geq \alpha$, i.e. C provides novel information with respect to the new set. If the local interestingness of C is greater than the one of R, it is better (for maximizing the interestingness) to select C and to remove R from the result set M. The new result set is then $M := (M \backslash R) \cup \{C\}$. Otherwise C is rejected and the set M remains unchanged.

By the incremental result construction we add only informative clusters to our set and additionally try to maximize the interestingness of all selected clusters.

Algorithm 3.2: Cluster selection procedure

```
1 ⟨C_1, ..., C_n⟩ := ranking of conceptGroup(C, M)
2 R := ∅ // clusters inducing the redundancy of C
3 N := ∅ // clusters not inducing the redundancy of C
4 for i:1...n do
5     if I_global(C, N ∪ {C_i}) ≥ α then N := N ∪ {C_i}
6     else R := R ∪ {C_i}
7 if I_local(C) > Σ_{C'∈R} I_local(C') then
8     add C to M and remove R from M
```

3.4.3 Efficient initialization

In general, full dimensional clusters rarely occur in high dimensional databases. If we started our top-down approach in full space we would analyze many uninteresting subspaces which are filtered out by our quality score criterion. For an efficiency boost we identify the topmost layer with interesting subspaces based on the idea of binary search. We start with subspace covering half of the dimensions (e.g. with 5d subspaces in a 9d database) and use our subspace search estimator to calculate the qualities. If we identify a subspace with sufficiently high quality we directly jump up to the dimensionality between the current subspace and the full space (e.g. from 5d to 7d). If no interesting subspaces are found we accordingly jump to lower dimensional spaces (e.g. from 5d to 3d). For this new dimensionality we repeat the "check-and-jump" procedure (with half the jump size) until we identify the highest dimensionality with interesting subspaces. This represents a binary search procedure.

In summary, our algorithm comprises three contributions to obtain a good approximation of the optimal orthogonal clustering. The binary search technique supports the top-down approach by an efficient initialization. The ranking of subspaces in a priority queue yields a preference of orthogonal and interesting subspaces. By recalculating the ranking, further subspaces can be pruned without clustering and novel concepts advance to the top of the priority queue. At last, the meaningful selection of new clusters to M results in an informative clustering since clusters with high interestingness values are chosen. In the next section we confirm this with an experimental analysis.

3.5 Experimental Analysis

We evaluate the quality and efficiency of the *OSCLU* approach compared to three variants of orthogonal clustering techniques (*Multi-View 1* and *Multi-View 2* proposed in [CFD07], and *Alternative Clustering* [QD09]), a recent non-redundant subspace clustering technique (*StatPC* [MS08]) and a projected clustering approach (*P3C* [MSE06]). For all algorithms we tried to find the optimal parameter settings for each data set.

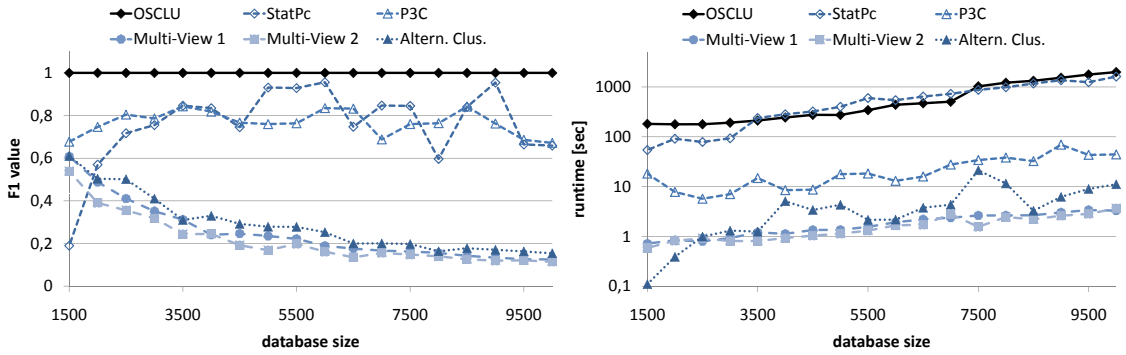
In general, we perform our evaluation on data with multiple hidden concepts. For both, synthetic and real world data, we extend single concept data as used for

the evaluation of traditional clustering approaches such that each object is part of multiple concepts. Thus, for a high quality clustering each object has to be detected in multiple clusters. While traditional clustering approaches are well suited for data with only one hidden concept, we compare our approach against recent techniques designed for multiple hidden concepts. For scalability experiments, we generate synthetic data following a method proposed in [KKK04, AKMS08a] to generate density-based clusters in arbitrary subspaces. In addition, our generator takes into account that objects can belong to multiple concepts. To obtain data with these characteristics, for each object we concatenated attribute values of different subspace clusters to a higher dimensional space with multiple hidden concepts per object. Further on, we show the performance of OSCLU on two extended real world data sets (original *iris* and *liver disorders* are provided by the UCI repository [FA10]). Again, we created multiple concepts per object by randomly concatenating objects of different classes, resulting in one high dimensional data set. As usual for the evaluation of clustering results, we use the class labels of the data as ground truth information.

To ensure comparability of evaluations, we measure runtimes on identical machines with 2.33GHz Intel XEON CPU, 2 GB of main memory and JAVA 1.6 runtime environment. For quality measurements, we use the $F1$ value [GFM$^+$11] that is used in evaluation of subspace and projected clustering [AKMS08a, MS08, MSE06, MGAS09].

3.5.1 Scalability

Database size. In Figure 3.4(a) we analyze the quality of the clustering results with respect to the database size. While increasing the number of objects, we keep the number of concepts fixed to three. We generate concepts with five relevant attributes such that overall we obtain a 15-dimensional data space. Our OSCLU algorithm yields the highest quality compared to all other algorithms, as we detect all hidden clusters in various concepts. The quality of OSCLU is independent of the database size and very robust. StatPC and P3C show good quality results, but also high fluctuating values that cannot reach the quality of OSCLU. All three orthogonal clustering approaches show only low and decreasing quality with respect to the database size. Their underlying k-means model tries to partition the data in each iteration of orthogonal cluster detection. Thus, it cannot cope with



(a) Quality of clustering (b) Runtime

Figure 3.4: Scalability w.r.t. database size

the fixed noise ratio in the data, which is always assigned to some of the detected clusters, hence, resulting in low quality clusterings.

The runtime with respect to the database size is presented in Fig. 3.4(b). The slopes of all curves are in the same range and the influence of the size on all algorithms becomes apparent. The two top quality approaches, our OSCLU model and StatPC, result in similar runtimes. Our redundancy checks and also our density-based model are very complex, but these aspects account for the high quality. The remaining algorithms are faster but we believe, that our runtime is still acceptable considering our high quality results. Especially with increasing concept number, as presented in the next experiment, our model outperforms all other approaches.

Number of concepts. The aim of our model is the detection of multiple concepts, which arise in real scenarios. Thus, in the next experiment we analyze the performance of the algorithm by increasing the number of concepts hidden in a database. To scale the number of concepts, we use a simple data set with only 1000 objects as most of the algorithms showed comparable quality values in this range in the previous experiment. We vary the number of hidden concepts in Figure 3.5 from 1 to 5.

As shown, OSCLU is able to detect clusters even if objects group in multiple concepts. It shows high quality even for a high number of hidden concepts. While traditional clustering approaches aim at clustering single concept data, the alternative clustering approach is designed for two concepts and the multi-view approaches should detect even more than two. However, even these approaches cannot compete with our model. For the subspace and projected clustering approaches, increasing number of concepts makes it very hard to detect the hidden

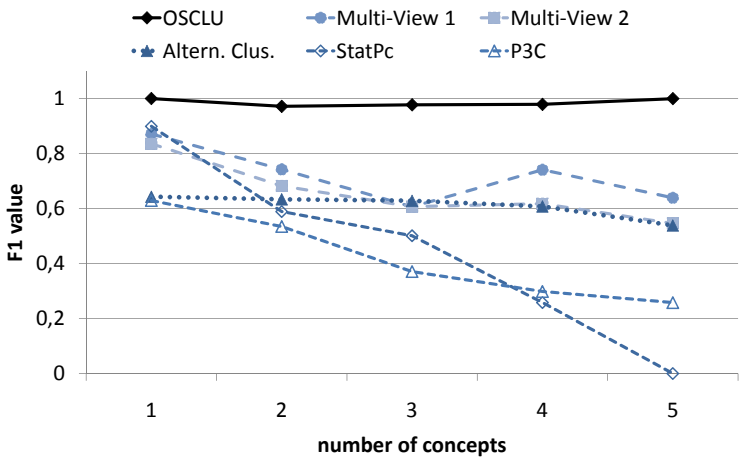


Figure 3.5: F1 vs. number of concepts

clusters. Especially the projected clustering approach P3C shows decreasing quality, as each object belongs to at most one concept. Overall, StatPC and P3C are not able to detect the multiple hidden concepts per object, while OSCLU yields very high clustering quality.

Noise percentage. In the previous experiments we showed that we outperform subspace and projected clustering approaches as they cannot cope with multiple hidden concepts. Thus, in the following experiments we focus on a more detailed comparison of OSCLU against the orthogonal clustering techniques detecting multi-view and alternative clusterings. First we analyze the effect of noisy data especially for high concept numbers. For the next experiment, illustrated in Figure 3.6, we generate data with five hidden concepts and vary the noise percentage. On such a difficult data setting, our OSCLU approach outperforms the other techniques. It can detect the clusters hidden in different concepts even in very noisy data sets. Both multi-view algorithms and the alternative clustering approach show again decreasing qualities.

3.5.2 Real world data

As we aim at the detection of multiple concepts, we focus our evaluation for real world data also on scalability w.r.t. number of concepts. We use single concept data obtained by UCI repository and extend them to multi concept data sets. As described in the experimental set-up, similar to synthetic data we can vary the complexity of data sets by including more and more hidden concepts. However,

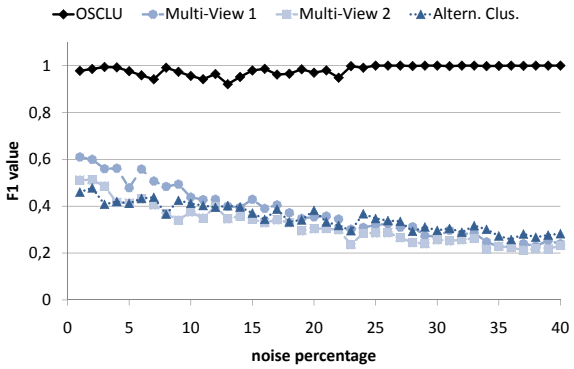
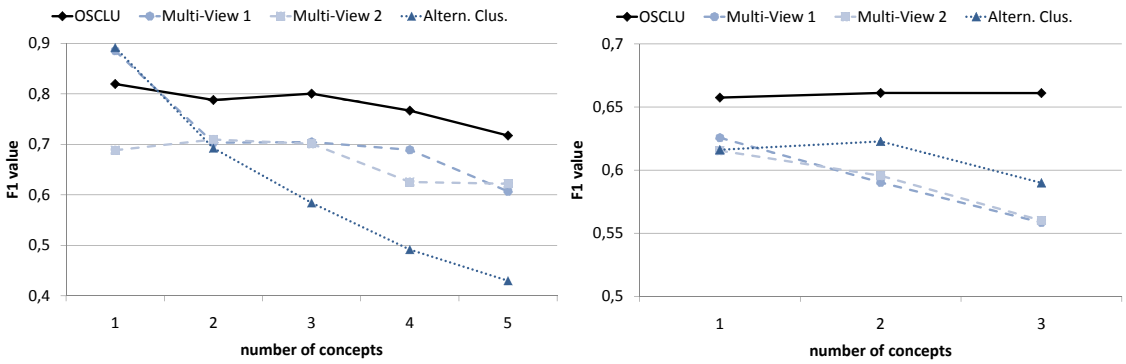


Figure 3.6: F1 vs. noise

in contrast to the previous experiments we use real world data distribution for the single concepts. We evaluate the effect of variable concept counts on the clustering quality, as for an increasing number of concepts it is more difficult for all algorithms to identify the hidden structure of the data.

In Figure 3.7 we show clustering quality on iris and liver disorder data set. For the very simple case of only one concept (original UCI data sets), the quality is high for all algorithms. However, for increasing number of hidden concepts, quality dramatically drops for all competing approaches. Especially, quality of the alternative clustering approach drops with more than two concepts, as it is designed for up to two concepts only. OSCLU shows significantly better performance as it keeps quality high, outperforming the competing approaches for multiple concept data. Only for the original iris data set one of the multi-view approaches shows higher quality values. Even though we parameterize the competing approaches



(a) Data set: multi-concept iris (b) Data set: multi-concept liver disorder

Figure 3.7: Quality on real world data sets with increasing number of concepts

that their resulting number of clusters matches the true number of hidden clusters, these approaches are not able to detect all hidden concepts. Overall, our OSCLU approach clearly outperforms all competing algorithms even for increasing number of concepts per object.

3.5.3 Parametrization

Additional to the experiments that compare OSCLU with existing methods, we analyze the flexibility of our model. Besides controlling the output of OSCLU depending on the application by varying the local interestingness function I_{local}, the user can also control the redundancy between the clusters. In Figure 3.8, we present the variation of the parameter α which controls the interestingness of each cluster in the result set. As intended by this parameter, higher values of α increase the required interestingness and, hence, less clusters are in the result, i.e. redundancy is evaluated more strict. By varying the α parameter one can control the overall result set based on our global interestingness I_{global}. We include a cluster only if the fraction of its newly clustered objects is at least α (cf. Definition 3.3).

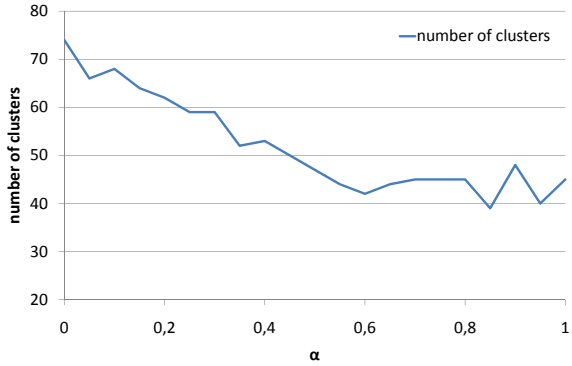


Figure 3.8: Variation of global interestingness

3.6 Conclusion

In this chapter we introduced the OSCLU (Orthogonal Subspace CLUstering) approach. It overcomes major limitations of existing approaches in the detection of multiple concepts hidden in arbitrary subspace projections of the data. Our clustering model detects multiple concepts per object. It computes an optimal orthog-

onal clustering by ensuring non-redundancy and maximal interestingness of the resulting clustering. We show that our clustering model is NP-hard and propose an efficient algorithm computing an approximate solution. We approximate the optimization problem by pruning similar subspaces ensuring efficient cluster detection since only orthogonal subspaces are analyzed. Thus, our OSCLU approach is the first method for detecting multiple clustering views in subspaces of high dimensional data. Thorough experiments demonstrate that OSCLU clearly outperforms existing subspace clustering and multi-view clustering algorithms while automatically reducing the output to only the orthogonal concepts hidden in the data.

Our OSCLU model provides a general solution to detect clusters hidden in multiple views, independent of the chosen subspace cluster definition. However, OSCLU does not explicitly mine the underlying views, i.e. it does not identify which clusters belong to a common view, and the views do not influence the individual groupings since they are solely determined through the cluster definition. In the following chapter, we propose a novel approach that overcomes these limitations by combining the cluster definition and the view detection into a single model.

Chapter 4

Multi-View Subspace Clustering via Multiple Mixture Models

Mixture models are a powerful method to represent and learn the clustering structure of data. Traditional mixture models, however, are limited to describe just a single grouping of the objects. In this chapter, we propose a novel generative model that accounts for data containing multiple clustering views. Our model represents the data's multiple views by different subspace projections, thus, avoiding the problem of full-space clustering. Each view describes an individual partitioning of the objects. Accordingly, our model is able to represent multiple groupings and it simultaneously prevents redundant information since highly overlapping clusters in similar subspace projections are avoided. Since views and clusters are located in subspace projections, we distinguish between relevant and irrelevant dimensions. Thus, unlike to traditional mixture model, in our model we have to tackle the challenge of model selection. Overall, our generative model couples the detection of views and their corresponding subspace clusters: the data's multiple clustering views are described by a single statistical model. To realize an efficient learning of this model, we develop the algorithm MVGen that utilizes the principle of iterated conditional modes. In thorough experiments, we show the effectiveness of our MVGen approach in comparison to other multi-view clustering methods.

4.1 Motivation and Comparison with Related Work

As discussed in the previous chapter of this thesis, each object can belong to various clusters. There does not exist only a single clustering solution but multiple groupings are possible, each providing different views on the data. Depending on the view, each object takes a certain role: in a customer database, for example, we might find the views 'hobby' and 'profession', and a specific person might take the roles of a 'soccer fan' and an 'architect'. Since clusters usually do not occur in the full-space, we have seen that views are characterized by subsets of the dimensions where the corresponding groupings can be identified (cf. Figure 3.1). Accordingly, also the detected clusters are hidden in subspace projections and some of their dimensions are irrelevant. Finally, as discussed in our OSCLU model, views can overlap in the sense that some dimensions might occur in multiple views. The attribute 'gender', for example, might be used to characterize several views, e.g. 'hobby' as well as 'profession'. Overall, as a consequence, datasets containing multiple clustering views are highly complex. Modeling such complex data in a single statistical model is a challenge not tackled so far.

A traditional way of modeling data with an underlying clustering structure is by using *mixture models*. Given a parametric family $\mathcal{K}$, e.g. the set of all Gaussian distributions, a mixture model describes the data by a set of components (each selected from $\mathcal{K}$) and a set of mixture weights. Intuitively, each component represents a cluster (more precisely: its distribution of the attribute values) and the mixture weight represent the number of objects belonging to this cluster. Since, overall, only a single mixture distribution is obtained, traditional mixture models are not suited to model multi-view data but are limited to a single view, i.e. each observation is assumed to follow a single component's distribution. Please do not confuse the notions of soft clustering and multi-view clustering. In soft clustering one describes the uncertainty about the decision which component an observation follows, while in multi-view clustering each object (certainly) follows multiple components. Thus, even if we apply the principle of soft clustering to traditional mixture models, i.e. we consider for each object the likelihood of belonging to a certain cluster, we do not realize multiple views. To put it simply, if the data shows a clear clustering structure, in traditional mixture models one typically observes for each object a high likelihood only for a single component, i.e. an object mostly belongs to a single cluster.

To overcome the last-mentioned issue, a few models [SJR10, SJR08, FB08,

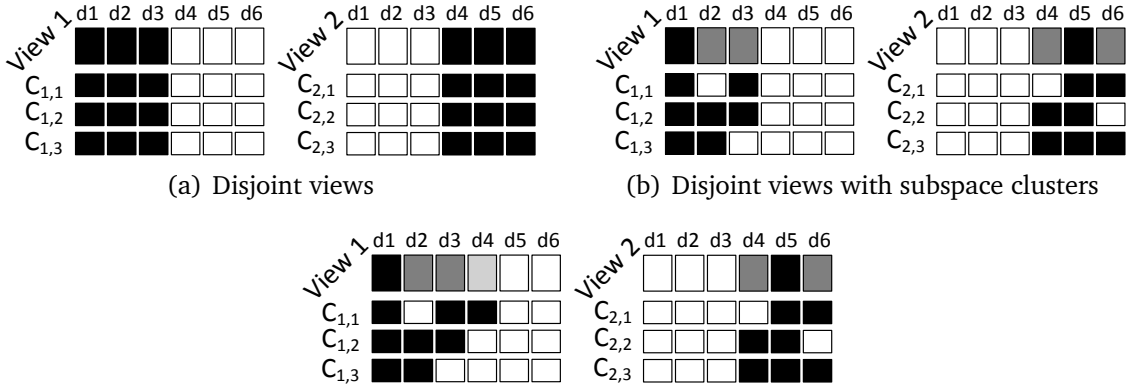
BKG+05] try to represent a mixed membership in different components, i.e. an object might follow multiple components. Thus, they realize an overlapping clustering and, e.g., allow for a movie to participate in the 'humor' as well as in the 'action' genre [BKG+05]. Though, these methods are still not suited for multi-view clustering. First, the existing works do not consider a grouping of clusters into views, i.e. they do not model that an object takes a single role within a single view but different roles among different views. Instead, these models lead to results where an object might take multiple roles within a single view. Therefore, these methods are able to represent only a single view on the data, e.g. the view 'genre' for the movie example. Different clusterings based, for example, on the movies' characteristics as 'location', 'cast', or 'budget' can not be represented. Second, the existing models are not able to model subspace clusters in a reasonable way, which, however, is a prerequisite for good clustering results as shown in the previous chapters of this thesis.

In the following we discuss the challenges for modeling multi-view data and we motivate how to represent these aspects in a novel statistical model. The advantage of such a statistical model based approach is twofold: First, describing multi-view data with a *single* generative model allows a tight coupling between the actual representation of views and the grouped subspace clusters. Second, learning such a model given a set of observed data objects directly leads to a method how to detect the data's multiple clustering solutions. Thus, in contrast to our OSCLU method, our novel clustering algorithm couples the detection of views and clusters: both aspects influence each other.

4.1.1 Challenges for modeling multi-view data

Instead of summarizing the data by a single mixture distribution, we generalize the data by using multiple mixture models. Taking a generative perspective, we can assume each object to be generated by *multiple mixture distributions*, each referring to a different view of the data. Consequentially, each object follows *multiple* components, each in a different mixture model, each defining a distribution only for a certain view (i.e. subspace) of the data, and each representing a different role of the object. This poses several challenges:

Challenge 1: Multiple Groupings. In the most simple case of multi-view data, multiple groupings are obtained by partitioning the dimensions of a database into non-overlapping views, and the relevant dimensions of each subspace cluster cor-



(a) Disjoint views

(b) Disjoint views with subspace clusters

(c) Overlapping views with subspace clusters

Figure 4.1: Different scenarios of multi-view data (black rectangles: relevant dimensions; white: irrelevant dimensions)

respond to its view's dimensions. That is, each cluster is a full-space cluster with respect to the view. A schematic representation of this case for a database with 6 dimensions, 2 views, each with 3 clusters, is given in Figure 4.1(a). The views' and clusters' relevant dimensions are depicted as black rectangles, while irrelevant dimensions are depicted in white. For example, the cluster $C_{1,1}$ is located in the subspace $\{d1, d2, d3\}$, while the cluster $C_{2,1}$ is located in $\{d4, d5, d6\}$. Intuitively, we can model this scenario by 'concatenating' several traditional mixture models. Each view represents an individual mixture model; thus, enabling different groupings among different views. To model this scenario, it is sufficient to provide, besides the actual components of the mixture models, an indicator variable that assigns each dimension to a corresponding view. In Figure 4.1(a), for example, the dimensions $\{d1, d2, d3\}$ are assigned to the first view.

Challenge 2: Subspace Clusters. Usually, the solution via traditional mixture models discussed above is too restrictive for the characteristics of real world data. While for a certain view a set of attributes is relevant in general, we cannot expect that each cluster covers exactly the same set of dimensions as its view (subspace cluster w.r.t. the view). While the dimension 'viewers age' might be relevant for the view 'genre' in general (e.g. 'Horror' movies target only adults), some genres like, e.g., '3D Animations' show no certain characteristic in this dimension. This scenario is illustrated in Figure 4.1(b).

Subspace clusters cannot be represented by traditional mixture models. While in the relevant dimensions of a subspace cluster, the attribute values are dis-

tributed according to, e.g., a Gaussian distribution, irrelevant dimensions follow a completely different model, e.g. a uniform distribution. That is, depending on the dimension's relevance a different parametric family is used. Thus, to model data containing subspace clusters, we encounter the challenge of model selection, i.e. before we can estimate the actual mixture model parameters, we first have to determine the parametric families that are used for each cluster.

Challenge 3: Overlapping Views. So far, we just discussed non-overlapping views. In general, however, dimensions can occur in multiple views (Fig. 4.1(c)). E.g., the dimensions 'gender' and 'age' might be characteristic for the two views 'hobby' and 'profession' of a customer data base. This scenario is particularly challenging as several components might compete with each other for generating an object in one or more dimensions (e.g. clusters $C_{1,1}$ and $C_{2,3}$ in dimension four). Obviously, the *dominant* view (and hence dominant distribution) might vary for each dimension and each object: while in dimension 'gender' some objects rate the view 'hobby' as dominant, other objects use the view 'profession' in this dimension; in the dimension 'age' completely different views might be considered as dominant. This observation is even intensified by considering subspace clusters: some clusters might not be relevant in the overlapping dimensions. Accordingly, it is not sufficient to consider only the views, but we need to consider the actual subspace clusters to determine the dominant view. Since each object might be located in different subspace clusters, different overlap scenarios can occur.

To tackle all these challenges, we propose in the next section a Bayesian framework modeling data with an inherent multi-view clustering structure. Our model:

- provides multiple generalizations of the data by modeling individual mixture models, each representing a distinct view
- handles individual sets of relevant dimensions for each cluster by performing Bayesian model selection
- tackles the ambiguity of the objects' memberships regarding multiple, competing components

4.2 Formal Description of the Model

In this section, we introduce a Bayesian framework modeling the process of generating data containing multiple clustering views. An overview of our framework

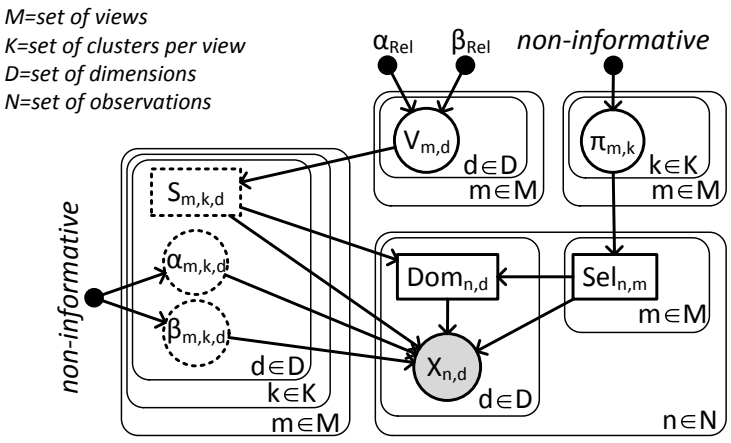


Figure 4.2: Graphical model of our method. Rectangles denote discrete random variables, circles continuous random variables, and black dots (deterministic) hyperparameters of the prior distributions.

is given by the graphical model depicted in Figure 4.2. We start in Section 4.2.1 by describing our model from a generative perspective, i.e. we show how our model generates data containing multiple views. The inverse process where a set of observations is *given* and the model's components are learned, is introduced in Section 4.2.2. In Section 4.2.3 we discuss the parameterization of our model.

Following convention, we do not distinguish between a random variable X and its realization $X = \mathbf{X}$ if it is clear from the context. As an abbreviation, we denote sets of random variables with the index $*$, e.g. $Y_{*,d}$ is the set of random variables $\{Y_{i,d}\}$ with i in the corresponding index domain, and Y is an abbreviation for the set $Y_{*,*}$.

4.2.1 Generating Multi-View Data

In our model we explicitly differentiate between the relevant dimensions of the clusters and the relevant dimensions of the views. The relevant dimensions of a view provide a concise description for the relevant dimensions for a *set* of clusters. That is, the clusters belonging to the same view are located in similar subspace projections. Since the clusters' relevant dimensions might slightly vary, the relevance of dimensions for the view can also vary. In Figure 4.1(b), for example, dimension 1 has a high relevance for view 1 since all of its represented clusters use this dimension; dimension 1 is a good descriptor for the whole set of clusters.

Dimension 3, in contrast, has a slightly smaller relevance since one of the clusters does not require this dimension. Thus, to reflect the differing relevances of dimensions for each view, our model includes the (continuous) latent variable $V_{m,d}$ on $(0, 1)$ for $m \in M = \{1, \ldots, m_{max}\}$ and $d \in D = \{1, \ldots, d_{max}\}$.

Based on this relevance information, the actual relevant dimensions of each subspace cluster can be generated. We model this aspect by the (discrete) random variable $S_{m,k,d}$ on $\{0, 1\}$ for $d \in D$, $k \in K$ and $m \in M$.[1] The latent variable is 1 if the dimension d is relevant for the k-th cluster of view m, and 0 otherwise. The higher the relevance of a view's dimension, the more likely is the dimension relevant for the cluster. This property can be realized by a Bernoulli process. With probability $V_{m,d}$ the dimension d is relevant for the cluster, while with probability $1 - V_{m,d}$ it is not. Formally, the distributions of the latent variables $S_{m,k,d}$ is given by

$$\begin{aligned} p(S_{m,k,d} = 1 \mid V_{m,d} = r) &= r \\ p(S_{m,k,d} = 0 \mid V_{m,d} = r) &= (1 - r) \end{aligned} \tag{4.1}$$

If the value of $V_{m,d}$ is either close to 1 or close to 0, then the subspaces of each cluster in the same view are very similar. If $V_{m,d}$ is close to 0.5, we do not have a clear decision, and thus the subspaces of the clusters may differ stronger.

Prior distributions. To allow a fully Bayesian approach, we specify prior distributions for the variables $V_{m,d}$. We select the prior according to a Beta distribution, i.e.

$$V_{m,d} \sim Beta(\alpha_{Rel}, \beta_{Rel})$$

with hyperparameters $\alpha_{Rel} \in \mathbb{R}_{>0}$ and $\beta_{Rek} \in \mathbb{R}_{>0}$. A Beta distribution is suited due to the following reasons: First, since $V_{m,d}$ simulates a Bernoulli process, the Beta distribution corresponds to its conjugate prior. Second, based on the hyperparameters, the user can control the views' purity. That is, one can control the similarity between the clusters' subspaces originating from the same view. As mentioned above, high similarity between the subspaces can be realized by choosing $V_{m,d}$ close to 1 or close to 0. This issue can be modeled by selecting $\alpha_{Rel} = \beta_{Rel} < 1$. If no knowledge about the views' purity is given, we can simply choose $\alpha_{Rel} = \beta_{Rel} = 1$, leading to a non-informative prior.

[1] To simplify our model description, we assume that each view describes k_{max} clusters, i.e. $K = \{1, \ldots, k_{max}\}$.

Generating the membership information. After generating the relevant dimensions of each cluster, we now aim at generating observations that follow multiple overlapping views. More precisely, in each of the views each object shall belong to a single cluster; thereby, we realize a single grouping within a single view and multiple overlapping groupings among different views.

This idea can be modeled by the latent variable $Sel_{n,m}$ on $K = \{1, \ldots, k_{max}\}$ that models which of the k_{max} clusters an object n follows in view m. The distribution of Sel is governed by the (relative) weights $\pi_{m,k}$ of the clusters, i.e.

$$p(Sel_{n,m} = k \mid \pi_{m,*}) = \pi_{m,k}$$

As for usual mixture models, the larger the weight of a cluster, the more objects belong (in expectation) to the cluster. Please note that in contrast to traditional mixture models, in our model each view represents a certain grouping of *all* objects. Thus, in our model we have $\sum_{k \in K} \pi_{m,k} = 1$ for each view $m \in M$, while in traditional mixture models the overall weight of all clusters is normalized to 1.

As discussed in challenge 3 (and illustrated in Fig. 4.1(c)), different views compete with each other. An object might belong to two clusters which both are marked as relevant in a specific dimension d. To solve the ambiguity about the object's membership in this dimension, we specify one of the views as dominant (for this object and dimension). This aspect is modeled by the latent variable $Dom_{n,d}$ on $M = \{1, \ldots, m_{max}\}$. Here, a view $m \in M$ can only be dominant in d if the selected cluster is also relevant in d. Thus, let $M'_{n,d} = \{m' \in M \mid S_{m', Sel_{n,m'}, d} = 1\}$ be the set of views that are potentially dominant for object n in dimension d, the distribution of $Dom_{n,d}$ is modeled by

$$p(Dom_{n,d} = m \mid Sel_{n,*}, S_{*,*,d}) = \begin{cases} 1/|M'_{n,d}| & \text{if } m \in M'_{n,d} \\ 0 & \text{if } m \notin M'_{n,d} \wedge M'_{n,d} \neq \emptyset \\ 1/|M| & \text{else} \end{cases}$$

That is, we randomly select a view from the potentially dominant views (case 1), while the remaining views cannot be selected (case 2). The third case just occurs, if none of the selected clusters of an object is relevant in this dimension. In this case, an arbitrary view can be selected as dominant since any cluster represents just noise in this dimension.

$$f(x;\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot x^{\alpha-1} \cdot (1-x)^{\beta-1}$$

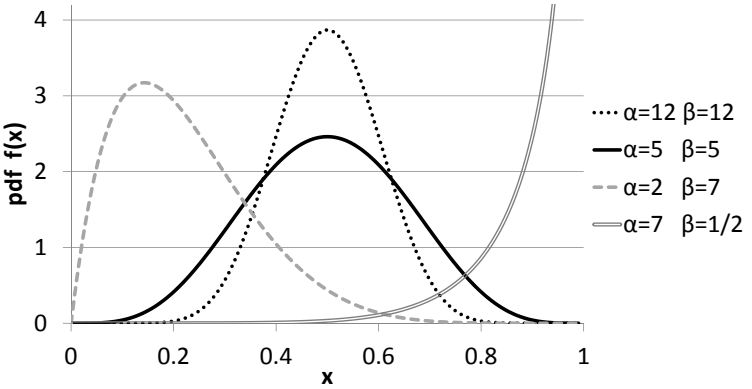


Figure 4.3: Probability density function of the Beta distribution

Generating Observations. Finally, we specify the distributions from which the attribute values of a cluster are sampled, i.e. we model the actual components of the multiple mixture models. Though, keep in mind that for subspace clustering we have two different parametric families: $\mathcal{K}_1$ for the relevant dimensions and $\mathcal{K}_0$ for irrelevant ones.

In our model we select the parametric family $\mathcal{K}_1$ according to the set of Beta distributions, i.e. we consider a mixture of Beta distributions. This is advantageous compared to the frequently used Gaussian distributions since Gaussian distributions have an infinite support, which usually does not match the observed data. In many applications, we have a finite attribute domain that can be normalized to the range $(0,1)$; this is exactly captured by the Beta distribution (cf. Figure 4.3). Additionally, the Beta distribution is able to model distributions near the border of the data space. These Beta distributions are modeled by the two random variables $\alpha_{m,k,d}$ and $\beta_{m,k,d}$ on $\mathbb{R}_{>0}$ providing the necessary shape parameters of each distribution (for each view m, each cluster k, and each dimension d). For the parametric family $\mathcal{K}_0$, we simply use the uniform distribution on $(0,1)$ since this corresponds to a noisy dimension. Thus, $|\mathcal{K}_0| = 1$ holds.

Which parametric family a mixture component in dimension d belongs to was modeled by the latent variable S. Thus, finally, the attribute values of each object can be modeled by the random variable $X_{n,d}$ with distribution:

$$X_{n,d} \mid Dom_{n,d}, Sel_{n,*}, S_{*,*,d}, \alpha_{*,*,d}, \beta_{*,*,d} \sim \begin{cases} Beta(\alpha_{i,j,d}, \beta_{i,j,d}) & \text{if } S_{i,j,d} = 1 \\ Uni(0,1) & \text{else} \end{cases}$$

where $Dom_{n,d} = i$ and $Sel_{n,i} = j$. That is, for each dimension d an object follows the distribution given by the *selected* cluster in the *dominant* view.

Prior distributions. Again, we choose appropriate prior distributions to enable inference. We select non-informative priors since usually no further knowledge is provided about the data's clustering structure. A non-informative prior for the cluster weights $\pi_{m,k}$ is simply realized by choosing $p(\pi_{m,1}, \ldots, \pi_{m,k_{max}}) = const$ for each view m.

For the variables $\alpha_{m,k,d}$ and $\beta_{m,k,d}$ a constant prior does not reflect this aspect and is anyway improper since the support of the variables is infinite. Thus, we suggest a non-informative prior $p(\alpha, \beta)$ that ensures a uniform distribution over the mean and variance of the resulting Beta distributions $Beta(\alpha, \beta)$. Intuitively, this way the cluster centers are uniformly selected from the domain $(0, 1)$ and the variance from the domain $(0, \frac{1}{12})$. Thus, the prior fulfills

$$\int_{\alpha}\int_{\beta} p(\alpha, \beta) \cdot \mathbf{1}(\mathbb{E}(Beta(\alpha, \beta)) = x) \, \mathrm{d}\alpha\mathrm{d}\beta \sim Uni(0, 1)$$

regarding the mean x of the resulting Beta distribution (same for the variance with $Uni(0, \frac{1}{12})$). We can approximately[2] achieve these properties by selecting the priors according to exponential distributions with rate parameter 0.1, i.e.

$$\alpha_{m,k,d} \sim Exp(0.1) \quad , \quad \beta_{m,k,d} \sim Exp(0.1)$$

4.2.2 Learning Objective

In the following, we describe our learning objective if a set of observed data points $\mathbf{X}$ is given. Usually, the learning objective would be to maximize the a posteriori probability $p(V, S, \alpha, \beta, Dom, Sel, \pi \mid X = \mathbf{X})$.

For our model, however, this idea is not meaningful since in this case usually all dimensions of a cluster are relevant: the data's likelihood is always higher when selecting a (certain) Beta distribution in contrast to selecting a uniform distribution. This is obvious since a uniform distribution is a special case of a Beta distribution with shape parameters $\alpha = \beta = 1$ and, hence, $\mathcal{K}_0 \subset \mathcal{K}_1$. Thus, simply determining the maximum a posteriori (MAP) estimate as given above leads to the

[2] Indeed, the distribution of the mean is exactly captured since the Beta distribution's mean is given by $\frac{\alpha}{\alpha+\beta}$, and for any λ it holds: $X, Y \sim Exp(\lambda) \Rightarrow \frac{X}{X+Y} \sim Uni(0, 1)$

problem of overfitting since a complex model obviously fits the data better than a simple one[3]; one would only choose relevant dimensions.

To overcome this problem, we first perform a model selection before learning the subspaces S and the shape parameters of the Beta distributions. That is, we balance the models' goodness of fit and their simplicity[4]. Thus, our learning objective is separated in two phases:

First, we perform Bayesian model selection [Bis06] by finding the best realization for V, Dom, Sel, and π. That is, we determine the MAP estimate

$$(\mathbf{V}^*, \mathbf{Dom}^*, \mathbf{Sel}^*, \boldsymbol{\pi}^*) = \arg \max_{(\mathbf{V},\mathbf{Dom},\mathbf{Sel},\boldsymbol{\pi})} p(V = \mathbf{V}, \\ Dom = \mathbf{Dom}, Sel = \mathbf{Sel}, \pi = \boldsymbol{\pi} \mid X = \mathbf{X})$$

These variables are illustrated in our graphical model with solid lines. Since learning these variables involves a marginalization over S, α, and β we realize the balancing of a model's complexity and its goodness of fit. Thus, due to this model selection step, some dimensions might be irrelevant for certain views, corresponding to a more simple model.

Since after the first phase the cluster model is determined, we can estimate in the second phase the actual mixture components and the clusters' subspaces. That is, we can now determine the MAP estimate for the variables S, α and β:

$$(\mathbf{S}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) = \arg \max_{(\mathbf{S},\boldsymbol{\alpha},\boldsymbol{\beta})} p(S = \mathbf{S}, \alpha = \boldsymbol{\alpha}, \beta = \boldsymbol{\beta} \mid X = \mathbf{X}, \\ V = \mathbf{V}^*, Dom = \mathbf{Dom}^*, Sel = \mathbf{Sel}^*, \pi = \boldsymbol{\pi}^*)$$

Overall, our model allows to learn the clustering structure of data containing multiple overlapping views by using multiple mixture models. Clusters, i.e. mixture components, are located in individual subspace projections and views summarize these clusters by a concise description of their relevant dimensions.

[3] A similar example is polynomial interpolation: since the set of polynomials with degree x is a subset of the ones with degree $x + 1$, the interpolation error decreases with increasing degree.

[4] In the example of polynomial interpolation, one balances the degree of the polynomial against its regression error.

4.2.3 Discussion of Parameters

Our model can be adapted based on four parameters, each of them easy to interpret: First, with M we specify the number of views our method should detect. Second, the parameter K controls the number of clusters per view. This is similar to the parameter as required by the k-means algorithm. Automatically determining these two parameters based on a given set of observations is a challenge on its own and should not be discussed here. Last, the two hyperparameters α_{Rel} and β_{Rel} control the prior distribution of the views. As mentioned above, if we expect a good grouping of the clusters into views one should select $\alpha_{Rel} = \beta_{Rel} < 1$. If no knowledge about the views is given, $\alpha_{Rel} = \beta_{Rel} = 1$ should be selected.

4.3 The MVGen Algorithm

In this section we introduce our MVGen (Multi-View Generative Model) algorithm that learns the multi-view clustering structure given a set of observed data points. Since exactly computing the MAP estimate $p(V, Dom, Sel, \pi \mid X)$ is intractable, we compute approximations that can be efficiently determined. In general, we exploit the principle of iterated conditional modes (ICM [Bes86]), which can be regarded as greedy variant of the Gibbs sampling method [Bis06]. That is, instead of considering a complex joint distribution $p(A_1, \ldots, A_n)$, we iteratively maximize a set of conditional probabilities $p(A_i \mid A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n)$ until the process converges. This way, the random variables A_i are updated sequentially. The traditional k-means processing scheme can be seen as an instance of the ICM principle with just two easily computable update steps: recomputation of means and reassignment of points to clusters.

4.3.1 Update Equations

In the following we will present the update equations required in our algorithm; they are summarized in Equation (U1)-(U4). For simplification, we first introduce two intermediate results.

BIC Approximation. During our derivation we have to evaluate the term

$$\int_{\alpha}\int_{\beta} p(\alpha)p(\beta)\prod_{n\in I} Beta(X_{n,d};\alpha,\beta)\ \mathrm{d}\alpha\mathrm{d}\beta$$

where I is an index set denoting which observations are actually considered.

Using the Bayesian Information Criterion (BIC, or Schwarz criterion [Sch78, Bis06]), and the observation that in our case the Beta distribution is controlled by two free parameters, we can approximate the above term by

$$Beta_d(I) := \left[\prod_{n\in I} Beta(X_{n,d}; \alpha_{MAP}, \beta_{MAP})\right] \;/\; |I| \tag{4.2}$$

where α_{MAP} and β_{MAP} are the MAP estimates of the shape parameters given the set of observations.

Reformulating the Likelihood. Let $N_{m,d} = \{n \in N \mid Dom_{n,d} = m\}$ be the set of all observations that choose the view m in dimension d as *dominant* and $N_{m,k,d} = \{n \in N_{m,d} \mid Sel_{n,m} = k\}$ be those observations which additionally *select* the cluster k. Obviously, for each dimension $d \in D$ we have $N = \bigcup_{m\in M} \bigcup_{k\in K} N_{m,k,d}$. Given the equation

$$\prod_{n\in N} \frac{V_{i(n),d}}{\sum_{m'\in M} V_{m',d}} \cdot p(X_{n,d} \mid Dom, Sel, S, \alpha, \beta)$$

where $i(n) = Dom_{n,d}$ denotes the dominant view of observation n. Instead of taking the product over each observation $n \in N$ individually, we can 'group' the observations according to their dominant view and selected cluster. That is, the above equation is equivalent to

$$\propto \prod_{m\in M} \left[\left(\frac{V_{m,d}}{\sum_{m'\in M} V_{m',d}} \right)^{|N_{m,d}|} \prod_{k\in K} \cdot \prod_{n\in N_{m,k,d}} \begin{cases} B_{k,d}^{n,m} & \text{if } S_{m,k,d} = 1 \\ 1 & else \end{cases} \right] \tag{4.3}$$

and $B_{k,d}^{n,m}$ is an abbreviation for $Beta(X_{n,d}; \alpha_{m,k,d}, \beta_{m,k,d})$.

Updating the views V. Using the above results, we now derive the update equation for the variable $V_{m,d}$, i.e. for each $m \in M, d \in D$ we aim at maximizing

$$p(V_{m,d} \mid V \backslash \{V_{m,d}\}, Dom, Sel, \pi, X) \tag{4.4}$$

$$\propto \textstyle\sum_S \int_\alpha \int_\beta p(V, Dom, Sel, \pi, S, \alpha, \beta, X)\, \mathrm{d}\alpha \mathrm{d}\beta$$

The most important aspect here is that we have to marginalize over the vari-

able S, α and β, as stated in Section 4.2.2. Only if the model selection step is performed, we can estimate α, β and S.

We will prove that the optimal realization of $V_{m,d}$ can be computed based on the equation

$$V_{m,d} = \arg\max_{x \in (0,1)} c_a \cdot \log x + c_b \cdot \log(1-x) + c_c \cdot \log(x + c_d) + \sum_{k \in K} \log(c_k \cdot x + 1) \tag{U1}$$

where the c_* are constant values given by

$$c_a = \alpha_{Rel} - 1 + |N_{m,d}| \qquad c_b = \beta_{Rel} - 1 \qquad c_c = -|N|$$

$$c_d = \textstyle\sum_{m' \in M, m' \neq m} V_{m',d} \qquad c_k = Beta_d(N_{m,k,d}) - 1$$

Overall, Equation (U1) describes a simple univariate function in the variable x whose optimization can, for example, be done by Brent's algorithm [Bre02].

The update equation can be derived as follows: Based on the dependencies given by our graphical model, maximizing Equation (4.4) is equivalent to maximizing

$$\propto \sum_S \int_\alpha \int_\beta p(V)p(S \mid V)p(\alpha)p(\beta)p(Dom \mid S, Sel)p(X \mid Dom, Sel, S, \alpha, \beta)\, \mathrm{d}\alpha \mathrm{d}\beta$$

Making the individual variables explicit, we obtain

$$\propto \sum_S \int_\alpha \int_\beta \prod_{d \in D} \left(\prod_{m \in M} \left[p(V_{m,d}) \prod_{k \in K} \left[p(S_{m,k,d} \mid V) p(\alpha_{m,k,d}) p(\beta_{m,k,d}) \right] \right] \cdot \right.$$
$$\left. \prod_{n \in N} \left[p(Dom_{n,d} \mid S, Sel) p(X_{n,d} \mid Dom, Sel, S, \alpha, \beta) \right] \right) \mathrm{d}\alpha \mathrm{d}\beta$$

Due to the summation over all possible realizations of the random variable S, the term $p(Dom_{n,d} \mid S, Sel)$ can well be approximated by the expected dominance of view $Dom_{n,d} =: i(n)$ in dimension d for observation n. The expected dominance is given by

$$\frac{V_{i(n),d}}{\sum_{m' \in M} V_{m',d}}$$

Thus, the above equation reformulates to

$$\propto \sum_S \int_\alpha \int_\beta \prod_{d \in D} \Bigg(\prod_{m \in M} \Big[p(V_{m,d}) \prod_{k \in K} \Big[p(S_{m,k,d} \mid V) p(\alpha_{m,k,d}) p(\beta_{m,k,d}) \Big] \Big] \cdot \underbrace{\prod_{n \in N} \Big[\frac{V_{i(n),d}}{\sum_{m' \in M} V_{m',d}} p(X_{n,d} \mid Dom, Sel, S, \alpha, \beta) \Big]}_{(*)} \Bigg) \, \mathrm{d}\alpha \mathrm{d}\beta$$

We now substitute part $(*)$ with the result obtained in Equation (4.3), and get after a reordering of the terms:

$$\propto \sum_S \int_\alpha \int_\beta \prod_{d \in D} \Bigg(\prod_{m \in M} \Big[p(V_{m,d}) \left(\frac{V_{m,d}}{\sum_{m' \in M} V_{m',d}} \right)^{|N_{m,d}|} \prod_{k \in K} \Big[p(S_{m,k,d} \mid V) \cdot \underbrace{p(\alpha_{m,k,d}) p(\beta_{m,k,d}) \prod_{n \in N_{m,k,d}} \begin{cases} B_{k,d}^{n,m} & \text{if } S_{m,k,d} = 1 \\ 1 & else \end{cases} \Big] \Big] \Bigg) \, \mathrm{d}\alpha \mathrm{d}\beta$$

The part $(**)$ can be substituted with the result obtained in Equation (4.2). Thus, the integration over α and β vanishes and the formula can further be simplified to:

$$\propto \sum_S \prod_{d \in D} \Bigg(\prod_{m \in M} \Big[p(V_{m,d}) \left(\frac{V_{m,d}}{\sum_{m' \in M} V_{m',d}} \right)^{|N_{m,d}|} \prod_{k \in K} \Big[p(S_{m,k,d} \mid V) \begin{cases} Beta_d(N_{m,k,d}) & \text{if } S_{m,k,d} = 1 \\ 1 & else \end{cases} \Big] \Big] \Bigg)$$

The summation over S disappears when making the two cases of $p(S_{m,k,d} \mid V)$ explicit, i.e. we introduce the two possible cases $p(S_{m,k,d} = 1 \mid V) = V_{m,d}$ and

$p(S_{m,k,d} = 0 \mid V) = 1 - V_{m,d}$. Thus, the formula simplifies to:

$$\propto \prod_{d \in D} \left(\prod_{m \in M} \left[p(V_{m,d}) \left(\frac{V_{m,d}}{\sum_{m' \in M} V_{m',d}} \right)^{|N_{m,d}|} \prod_{k \in K} \Big[V_{m,d} \cdot Beta_d(N_{m,k,d}) + (1 - V_{m,d}) \cdot 1 \Big] \right] \right)$$

Let $c_d = \sum_{m' \in M, m' \neq m} V_{m',d}$. Since for the update of $V_{m,d}$ all remaining $V \backslash \{V_{m,d}\}$ are fixed, we have to maximize for each $V_{m,d}$:

$$p(V_{m,d}) \prod_{m' \in M} \left(\frac{V_{m',d}}{c_d + V_{m,d}} \right)^{|N_{m',d}|} \cdot \prod_{k \in K} [V_{m,d} \cdot Beta_d(N_{m,k,d}) + (1 - V_{m,d}) \cdot 1]$$

Note that we still have to consider the terms $\left(\frac{V_{m',d}}{c_d + V_{m,d}} \right)$ for all $m' \in M$ since the variable $V_{m,d}$ appears in the denominator. Maximizing the above equation is equivalent to maximizing

$$\propto p(V_{m,d}) \cdot V_{m,d}^{|N_{m,d}|} \cdot \prod_{m' \in M} (c_d + V_{m,d})^{-|N_{m',d}|} \cdot \prod_{k \in K} [V_{m,d} \cdot (Beta_d(N_{m,k,d}) - 1) + 1]$$

Since $\bigcup_{m' \in M} N_{m',d} = N$ and $V_{m,d}$'s prior follows a Beta distribution, we get

$$\propto V_{m,d}^{\alpha_{Rel}-1} \cdot (1 - V_{m,d})^{\beta_{Rel}-1} \cdot V_{m,d}^{|N_{m,d}|} \cdot (c_d + V_{m,d})^{-|N|} \cdot \prod_{k \in K} [V_{m,d} \cdot (Beta_d(N_{m,k,d}) - 1) + 1]$$

Taking the logarithm of the above equation and substituting $V_{m,d}$ with x yields

$$\propto (\alpha_{Rel} - 1 + |N_{m,d}|) \cdot \log x + (\beta_{Rel} - 1) \cdot \log(1 - x) + (-|N|) \cdot \log(x + c_d) + \sum_{k \in K} \log((Beta_d(N_{m,k,d}) - 1) \cdot x + 1)$$

Overall, the above equation corresponds to the update Equation (U1), which leads to a new realization for $V_{m,d}$.

$$V_{m,d} = \arg \max_{x \in (0,1)} c_a \cdot \log x + c_b \cdot \log(1 - x) + c_c \cdot \log(x + c_d) + \sum_{k \in K} \log(c_k \cdot x + 1)$$

where the c_* are constant values given by

$$c_a = \alpha_{Rel} - 1 + |N_{m,d}| \qquad c_b = \beta_{Rel} - 1 \qquad c_c = -|N|$$
$$c_d = \textstyle\sum_{m' \in M, m' \neq m} V_{m',d} \qquad c_k = Beta_d(N_{m,k,d}) - 1$$

Updating π. We perform a block update for the variables $\pi_{m,*}$. Since we use a non-informative prior, maximizing $p(\pi_{m,*} \mid V, Dom, Sel, \pi \backslash \{\pi_{m,*}\}, X)$ is simply obtained by

$$\pi_{m,k} = |\{n \in N \mid Sel_{n,m} = k\}| \cdot |N|^{-1} \quad \forall k \in K \tag{U2}$$

Updating Dom and Sel. Finally, we derive the update equations for the variables Dom and Sel. We perform a block update of the variables $Sel_{n,m}$ and $Dom_{n,*}$, i.e. for each observation n we simultaneously update its selected cluster in view m and its dominant views over all dimension. Formally, we aim at maximizing:

$$p(Sel_{n,m}, Dom_{n,*} \mid V, Dom \backslash \{Dom_{n,*}\}, Sel \backslash \{Sel_{n,m}\}, \pi, X)$$

$$\propto \sum_S \int_\alpha \int_\beta p(S \mid V) p(Sel_{n,m} \mid \pi) \prod_{d \in D} \Big[p(Dom_{n,d} \mid S, Sel) \\ p(\alpha) p(\beta) \prod_{n' \in N} p(X_{n',d} \mid Dom, Sel, S, \alpha, \beta) \Big] \, \mathrm{d}\alpha \mathrm{d}\beta \tag{4.5}$$

We first resolve the integral over α and β by again using the BIC approximation (cf. Eq. 4.2). We assume that the MAP estimates α^{MAP} and β^{MAP} derivable from the current grouping change only marginally when reassigning a single point n to a different cluster. Similarly, the cluster sizes change only marginally, i.e. the sets $N_{m,k,d}$ differ by at most one element when reassigning observation n to a different cluster. Using this idea, we can substitute the part $p(\alpha)p(\beta) \prod_{n' \in N} p(X_{n',d} \mid Dom, Sel, S, \alpha, \beta)$ by $Beta(X_{n,d}; \alpha^{MAP}_{i,j,d}, \beta^{MAP}_{i,j,d})$ if $S_{i,j,d} = 1$ and by 1 (uniform distribution) if $S_{i,j,d} = 0$.[5] This simplification stems from the fact that with given, constant MAP estimates also the densities $p(X_{n',d} \mid ...)$ for $n' \neq n$ are constant.

[5] We use the abbreviations $i{:=}Dom_{n,d}$ and $j{:=}Sel_{n,Dom_{n,d}}$

Thus, Equation 4.5 simplifies to:

$$\propto \sum_{S} p(S \mid V) p(Sel_{n,m} \mid \pi) \prod_{d \in D} \Big[p(Dom_{n,d} \mid S, Sel) \begin{cases} Beta(X_{n,d}; \alpha_{i,j,d}^{MAP}, \beta_{i,j,d}^{MAP}) & \text{if } S_{i,j,d} = 1 \\ 1 & else \end{cases} \Big] \quad (4.6)$$

Due to the integration over all possible realizations of S, the term $p(Dom_{n,d} \mid S, Sel)$ can well be approximated by the expected dominance of a view m in dimension d. The expected dominance is given by

$$ED_d(m) := \frac{V_{m,d}}{\sum_{m' \in M} V_{m',d}}$$

Thus, the above equation simplifies to

$$\propto \sum_{S} p(S \mid V) p(Sel_{n,m} \mid \pi) \prod_{d \in D} \Big[ED_d(Dom_{n,d}) \begin{cases} Beta(X_{n,d}; \alpha_{i,j,d}^{MAP}, \beta_{i,j,d}^{MAP}) & \text{if } S_{i,j,d} = 1 \\ 1 & else \end{cases} \Big] \quad (4.7)$$

Since the $S_{m,k,d}$ are independent given V, the summation over S is effective only for the variable $S_{i,j,d}$. Thus, the summation vanishes when making the two cases of $S_{i,j,d}$ explicit. That is, we introduce the term

$$AD_d(m,k) := p(S_{m,k,d} = 1 \mid V) \cdot Beta(X_{n,d}; \alpha_{m,k,d}^{MAP}, \beta_{m,k,d}^{MAP}) + p(S_{m,k,d} = 0 \mid V) \cdot 1$$

and Equation 4.7 simplifies to

$$p(Sel_{n,m} \mid \pi) \cdot \prod_{d \in D} ED_d(Dom_{n,d}) AD_d(Dom_{n,d}, Sel_{n,Dom_{n,d}})$$

Please note that the functions ED_d and AD_d are independent of Sel and Dom and fully specified by the values of V, α^{MAP}, and β^{MAP} (which are given!). Thus, while updating the values of Dom and Sel, we do *not* have to recompute the functions ED_d and AD_d.

Based on the above equation it becomes apparent that $Dom_{n,*}$ can be optimized

for each dimension individually. Especially, if the variable Sel is *given*, we can efficiently compute the optimal realization of $Dom_{n,d}$ by

$$Dom_{n,d} = \arg\max_{m \in M} ED_d(m) AD_d(m, Sel_{n,m}) \tag{U3}$$

As shown, the optimal realization of Dom depends on Sel in a simple way. Thus, we can focus on finding a good solution for $Sel_{n,m}$. The optimal solution of $Sel_{n,m}$ can efficiently be computed by

$$Sel_{n,m} = \arg\max_{k \in K} \left\{ \pi_{m,k} \prod_{d \in D} \max\left\{ ED_d(m) \cdot AD_d(m,k), c_{m,d} \right\} \right\} \tag{U4}$$

where $c_{m,d} = \max_{m' \in M, m' \neq m} ED_d(m') \cdot AD_d(m', Sel_{n,m'})$. The value of $c_{m,d}$ is constant since it neither depends on $Sel_{n,m}$ nor on $Dom_{n,*}$. Please note that the update of $Sel_{n,m}$ directly uses the best solution for $Dom_{n,*}$. Thus, we do not have to optimize $Dom_{n,*}$ separately but the optimal values are computed while updating $Sel_{n,m}$.

4.3.2 Recommended Update Sequence

Given the derived update equations, any sequence that recurrently invokes each of these equations is possible to determine a desired clustering solutions. However, based on the dependencies as given by our graphical model and the particular role of the model selection phase, we recommend the following update sequence for the random variables:

1. We sequentially update the variables $V_{m,d}$ for each $m \in M, d \in D$ until the views are stable [Eq. (U1)]
2. For each object $n \in N$, we sequentially update the variables $Sel_{n,*}$ and $Dom_{n,*}$ until the variables Sel and Dom are stable
   (a) To update $Sel_{n,*}$ and $Dom_{n,*}$ we sequentially update $Sel_{n,m}$ for each $m \in M$ until $Sel_{n,*}$ is stable. [Eq. (U3) & (U4)]
3. Update of $\pi_{m,*}$ for each $m \in M$ [Eq. (U2)]; goto step (2) until the process has converged
4. goto step (1) until the process has converged

Algorithm 4.1: The MVGen algorithm

```
1  initialization // cf. Section 4.3.4
2  while not converged do
3      while not converged do // update of views
4          for each m ∈ M, d ∈ D do update of V_{m,d} // Equation (U1)
5      while not converged do
6          for each object n ∈ N do // update Sel_{n,*} and Dom_{n,*}
7              while not converged do
8                  for each view m ∈ M do
9                      determine best value for Sel_{n,m} // Equation (U4)
10                     // Dom_{n,*} is automatically set by Equation (U3)
11         update π // Equation (U2)
12 determine α, β, and S // cf. Section 4.3.3
```

This update sequence is illustrated in the pseudo code of Algorithm 4.1. Thus, overall, we exploit the ICM principle in a nested fashion. The outer loop iterating over steps 1 and 2/3 represents the alternation between learning the views and learning the groupings. For the inner loop, iterating over 2 and 3, the views are given and we try to optimize the cluster assignments as good as possible. Note that implicitly also the mixture components α and β are optimized since based on the BIC approximation their MAP estimates are considered.

4.3.3 Determining Components and Subspaces

According to Section 4.2.2, the second phase of our learning objective is to determine the MAP estimate of $p(S, \alpha, \beta \mid X, V, Dom, Sel, \pi)$.

Since the variables Dom and Sel are given, the set of observations that contribute to the Beta distribution of cluster k in dimension d and view m is known, and was denoted by the set $N_{m,k,d} = \{n \in N_{m,d} \mid Sel_{n,m} = k\}$. Thus, the shape parameters $\alpha_{m,k,d}$ and $\beta_{m,k,d}$ of each mixture component simply correspond to their MAP estimate given the set of observations $N_{m,k,d}$.

In general, however, determining the MAP estimate for the shape parameters of a Beta distribution is not possible in closed form; one has to iteratively solve systems of equations [BT78]. Since this is highly inefficient, we refer to the commonly used approach of moment matching: the shape parameters are computed

based on the mean and variance values of the observations. This approach is in line with the non-informative prior distributions of α and β, which do not favor certain means or variance values. Thus, we get:

$$\alpha_{m,k,d} = \frac{(1-\mu)\mu^2 - \mu \cdot \sigma^2}{\sigma^2} \quad \text{and} \quad \beta_{m,k,d} = \frac{\mu(\mu-1)^2}{\sigma^2} + \mu - 1$$

where μ denotes the mean of the observations $\{X_{n,d} \mid n \in N_{m,k,d}\}$, and σ^2 the variance, respectively. Note that these equations are also used for the MAP estimates required in Section 4.3.1. Thus, the estimates for α^{MAP} and β^{MAP} can be efficiently computed in these steps.

Finally, an estimate for the random variables S can be obtained by testing which of the models – relevant or irrelevant dimension – is more likely. That is, if

$$p(S_{m,k,d} = 1 \mid V) \cdot \prod_{n \in N_{m,k,d}} Beta(X_{n,d}; \alpha_{m,k,d}, \beta_{m,k,d}) > p(S_{m,k,d} = 0 \mid V) \cdot 1$$

holds, the dimension d of cluster k in view m will be relevant. Here, we do not have to refer to the BIC approximation but can use the likelihood of the Beta distribution. Since the view V is already learned, i.e. the model selection is done, the trade-off between the models' complexities and their goodness of fit is already reflected in the term $p(S_{m,k,d} \mid V)$.

4.3.4 Initialization

To complete the overall algorithm, we describe a straight-forward initialization. We simply initialize Sel by the following method: For each dimension $d \in D$ we apply the k-means method with $k = |K|$. Thus, leading to $|D|$ many object groupings. Since, however, Sel requires just $|M|$ different groupings, we follow an approach inspired by traditional agglomerative clustering methods: To reduce the number of groupings, we successively determine those groupings that are most similar to each other, and we merge these groupings to a single one. Thus, in each step the number of groupings is reduced by one until the required number $|M|$ is reached. To merge two groupings, we simply union the corresponding sets of dimensions and we recompute the k-means result in the novel space. As similarity measure between the groupings we use the F-measure [WXC09].

After initializing Sel, the variable π is determined based on Equation (U2).

Furthermore, Dom is initialized randomly since no information about the views is given. The variable V is also initialized randomly based on its prior distribution.

4.4 Experimental Analysis

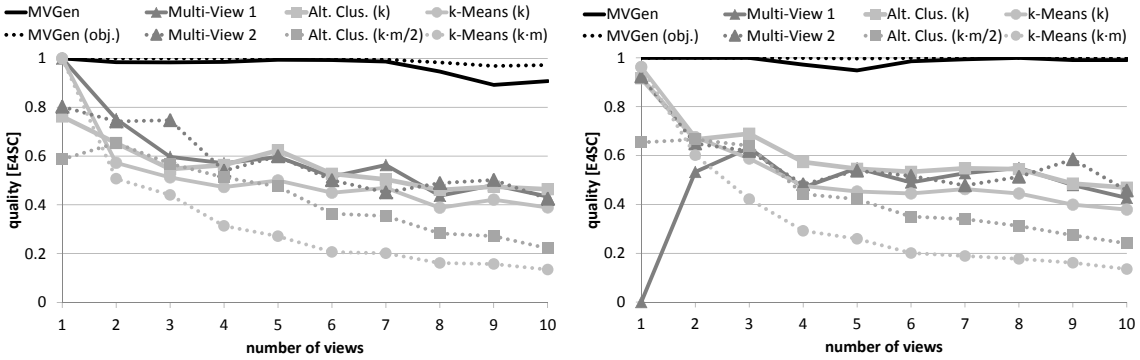
4.4.1 Experimental setup

Since our MVGen is a partitioning clustering technique (per view), we compare it with the multi-view clustering techniques *Multi-View 1* and *Multi-View 2* proposed in [CFD07], with two variants of the *Alternative Clustering* method proposed in [QD09], and two variants of the k-means methods. All of these techniques perform also a partitioning of the objects per view.

The methods of [CFD07] and our MVGen are provided for each data set with the true number m_{max} of views and the true number k_{max} of clusters per view. Since the method of [QD09] just detects two groupings, we use two variants: First, we use the method with the true number of clusters per view. In this case the method detects $2 \cdot k_{max}$ clusters. Second, we parametrize the method with $k_{max} \cdot m_{max}/2$. In this case, the method detects $k_{max} \cdot m_{max}$ clusters, which matches the overall number of clusters hidden in the dataset. Similar, we use for the k-means method two variants: first, we use as k the number of clusters per view, i.e. $k = k_{max}$; second, we use as k the number of overall clusters in the dataset, i.e. $k = k_{max} \cdot m_{max}$.

We generate synthetic data containing multiple views based on our generative model. The default data set contains 2 disjoint views, each with 5 clusters, 10 dimensions, 1000 objects, and the view's purity is 0.95. We vary the different characteristic of the data in our experiments. For case studies on real world data we use the CMUFaces, liver disorders, diabetes, iris and vowel data (all from the UCI repository [FA10]), and Escher images.

To ensure comparability of evaluations, we measure runtimes on identical machines with 2.33GHz Intel XEON CPU and 8 GB of main memory. For quality assessment we use the $E4SC$ measure [GFM+11] that is used in evaluation of subspace clustering. Since, however, our MVGen is the only method that performs multi-view subspace clustering, we ignore the subspaces for the competing methods in the evaluation. That is, these methods already get a huge advantage since they are implicitly provided with the correct subspace projections. Thus, for a fair



(a) constant data dimensionality (b) varying data dimensionality

Figure 4.4: Clustering quality w.r.t. a varying number of views

comparison, we also include the results of MVGen if we ignore the subspaces in the evaluation, denoted with 'MVGen (obj.)' since only the object groupings are important.

4.4.2 Evaluation of Clustering Quality

In our work we aim at detecting the clustering structure of data containing multiple views. Thus, we start our evaluation in Figure 4.4(a) by increasing the number of hidden views in the data set. In this experiment, we keep the overall dimensionality of the data fixed to 30, i.e. while the number of views increases, their average dimensionality decreases. As depicted, MVGen is the only approach able to detect the clustering structure in the case of a large number of views. MVGen is robust with respect to the number of views in the data. The clustering quality is very high, even if we incorporate the detected subspaces in our evaluation (solid line of MVGen). Obviously, the clustering quality is even higher if we just evaluate the object groupings (dashed line). The competing methods show a a different behavior: while for single-view data the quality of almost all approaches is high, their quality heavily decreases with an increasing number of views. Interestingly, for a high number of views, the quality of the two multi-view techniques (depicted by triangles) is not much larger than the quality of the k-means method with $k = k_{max}$. Thus, these methods are not suited to analyze data containing multiple views.

To ensure that the previous observations are due to the increasing number of

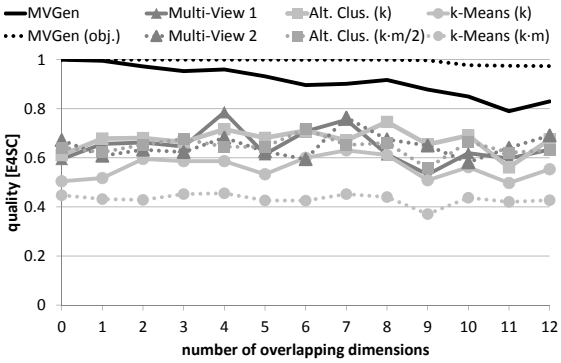


Figure 4.5: Effects of overlapping views

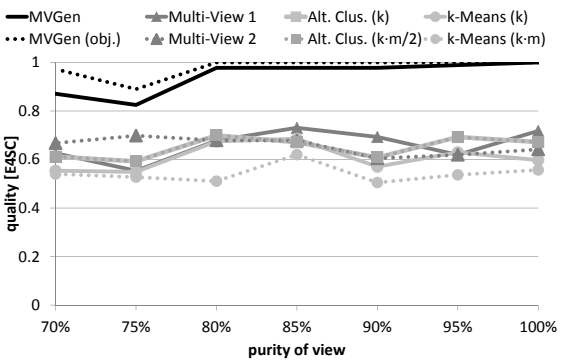


Figure 4.6: Effects of the views' purity

views and not due to the decreasing average dimensionality of the views, we perform a second experiment: In the experiment depicted in Figure 4.4(b), we again increase the number of views in the data; now, however, we keep the average dimensionality of the views fixed to 3. Thus, while increasing the number of views we also increase the overall dimensionality of the dataset. As shown, we get similar results: MVGen obtains high clustering qualities, while the competing methods show decreasing quality values. The competing methods cannot handle data sets with a large number of views. Please note that the Multi-View 1 approach is not applicable for the first data set: the method is not capable to handle data where the number of clusters is larger than the data's dimensionality. This, however, is problematic for multi-view data since obviously each view might contain an arbitrary number of clusters.

In the next experiment, we analyze the potential of our method to detect overlapping views. In Figure 4.5 we use a dataset with 12 dimensions containing 3 views. We vary the number of overlapping dimensions, i.e. dimensions that occur in more than one view, until at the end each dimension occurs in two views (12 overlapping dimensions in a data set with 12 dimensions). As shown, the methods are nearly not influenced by overlapping dimensions. The reason might be that none of the views is completely contained in another one. One is still able to detect the clusters of each view. While the object groupings are nearly perfectly recovered by MVGen, some of the clusters' relevant dimensions are missed for high overlapping degrees (indicated by the slightly decreasing solid line, and an almost constant dashed line). Note that the good quality of the competing approaches can only be observed since in this experiment just 3 views are used.

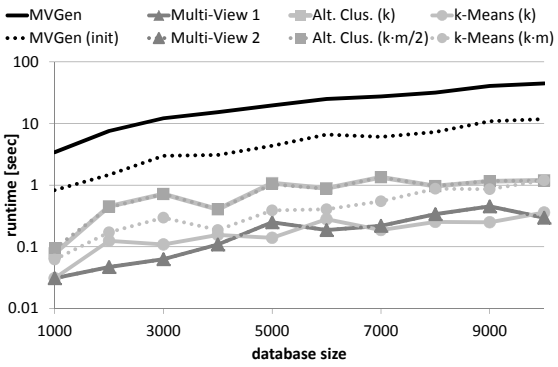


Figure 4.7: Scalability w.r.t. the database size

In Figure 4.6 we show that MVGen is able to find subspace clusters (located in subspace views). In our model we allow a certain deviation of the clusters' subspaces to the relevant dimensions of their views, denoted as the purity of a view. In this experiment, we vary the purity from 70% to 100%. Since in this experiment each view covers 5 dimensions, a purity of 70% already leads to subspaces clusters of dimensionalty 3, i.e. we get two additional irrelevant dimension per cluster. As show, MVGen suceesfully detects the relevant dimension of each cluster. Since we use a model selection approach, we trade-off the simplicity of the model against its goodness of fit. For the competing approaches we cannot draw any conclusions since for their evalution we do not consider the detected subspaces.

4.4.3 Evaluation of Scalability

Even though the focus of our method is to obtain high clustering quality, we briefly analyze its efficiency. We start by increasing the number of objects in the database. As depicted in Figure 4.7, all methods show increasing runtime and the slopes of the curves are in a similar range. Please note that the two approaches Multi-View 1 & 2 have almost identical runtimes, and, since we use 2 views, the two Alt. Clus. approaches are also identical in their runtimes. Apparently, the absolute runtime of our method is the highest. This is explainable by two facts: First, all competing approaches are based on the k-means method and they apply this method at most m_{max} times. In our initialization, however, we apply k-means already $d + (d - m)$ times. The runtime of this initialization phase is indicated in the figure by the dashed line, and it constitutes in this experiment to most of the overall runtime of our algorithm (solid line). Second, our model selection phase

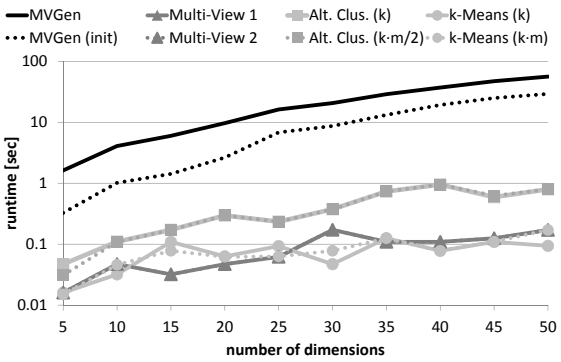


Figure 4.8: Scalability w.r.t. the data dimensionality

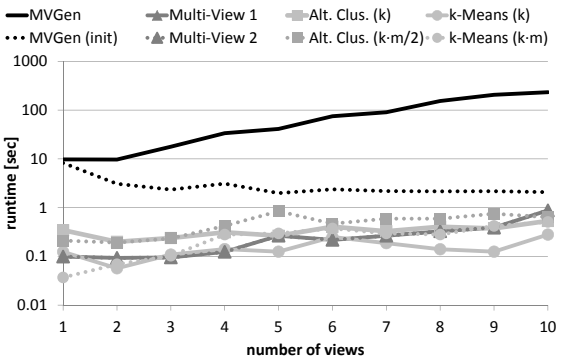


Figure 4.9: Scalability w.r.t. the number of views

that trades off relevant and irrelevant dimensions is complex and is not done by the competing methods. Even though, the remaining algorithms are faster, the absolute runtime of MVGen is still low. Furthermore, as we believe, the higher runtime is compensated by the significantly higher clustering quality of MVGen.

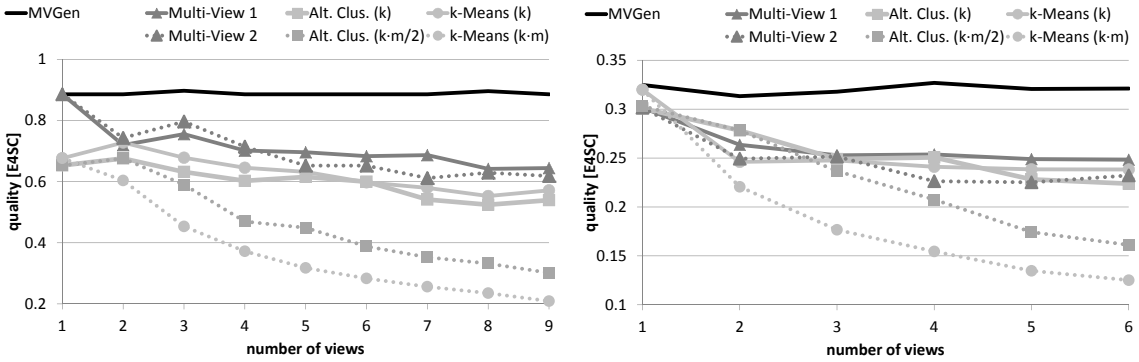
In Figure 4.8 we increase the dimensionality of the data set. We observe a similar behavior as in the previous experiment. Again, the initialization phase is responsible for most of the overall runtime. With increasing dimensionality this effect is even more amplified. Since we just use two views, the actual effort for the model selection phase is relatively small (please note the logarithmic scale of the y-axis).

Finally, we analyze the efficiency of MVGen if the number of views in the data set is increased. In Figure 4.9 the the number of views increases up to 10 while the dimensionality of the data is fixed to 30. As shown, this leads to an almost constant initialization phase. The actual runtime of the model selection phase, however, now increases since detecting the views and subspace clusters gets more challenging. The runtime of the initialization phase can be neglected for a high number of views (again note the logarithmic scale).

Overall, MVGen shows good scalability and it is the only method that simultaneously achieves high clustering qualities.

4.4.4 Real world data

For evaluation on real world data we use different evaluation principles, all focusing on the aspect of detecting multiple views. In our first experiment, we extend



(a) multi-view iris data (b) multi-view vowel data

Figure 4.10: Clustering quality on real world data

the datasets iris and vowel to data containing multiple views: for this, we randomly concatenate the attribute values of different objects to a higher dimensional space. The original data sets have dimensionalities of 4 and 10, respectively, while the extension to multi-view data leads to dimensionalities up to $9 \cdot 4 = 36$ (iris) and $6 \cdot 10 = 60$ (vowel), respectively. Figure 4.10 shows the results: For a small number of views, the quality of some competing approaches is similar to the one of MVGen. However, increasing the number of views leads to decreasing clustering quality for all competing approaches. In contrast, MVGen shows constant quality values; MVGen is not affected by an increasing number of views but detects the different object groupings even for a high number of views. These results for real world data are consistent with the observations made for the synthetic data.

In the next experiment we analyze the clustering result of MVGen on the CMU-Face data. This data is interesting for multi-view clustering since it consists of images taken from persons showing varying characteristics as their facial expressions (neutral, happy, sad, angry), head positions (left, right, straight), and eye states (open, sunglasses). As also done in [DB10a], we randomly select 3 persons with all their images and applied PCA retaining at least 90% of the data's variance as a pre-processing step. The clustering result of MVGen for two views each with three clusters is illustrated in Figure 4.11. The images correspond to the means of each detected cluster. By visual inspection, we can easily find the reason for detecting these two views: The first view, describes the grouping based on the different persons, while the second view, corresponds to a grouping based on their head positions.

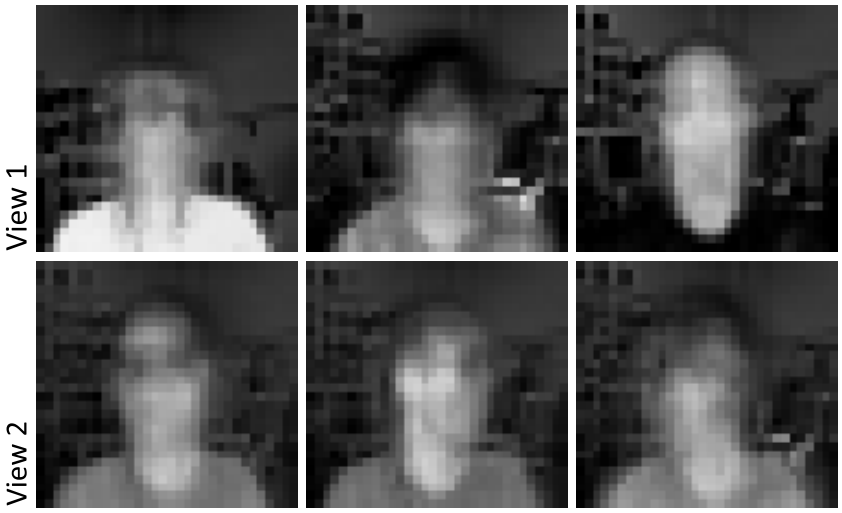


Figure 4.11: MVGen on face data

Next, we perform an experiment as introduced in [QD09]. They propose to perform image segmentation on Escher images, which are known to have multiple interpretations to the human eye. For clustering, each pixel is regarded as an object with RGB and HSV values as features. In Figure 4.12 (top), such an image is depicted (followed by the three views detected by MVGen). Focusing on the dark regions, there is a segmentation of the image as given by the first view of MVGen. This segmentation is dominant since the dark parts clearly deviate from the orange/yellow parts. However, MVGen is also able to discover the more subtle view where the yellow parts are decoupled from the others. Most interesting is the third view detected by MVGen: it corresponds to only the background of the image. For the competing methods we observed the following: The work of [QD09] was only able to detect groupings similar to MVGen's first and second view (as also shown in [QD09]). Interestingly, the work of [CFD07], which is designed to detect more than 2 views, was only able to find view 1. The detected 'alternative' groupings were all similar. None of the competing methods was able to detect the third, background view.

In our last case study, we want to highlight the benefit of explicitly modeling the relevant subspace for each view as done by MVGen. Knowing the relevant attributes enables us to reason about the views context and to explain the clusters. Table 4.1 depicts for the liver disorders and diabetes data the detected subspaces of each view. The number of clusters per view was chosen as 2. As shown for

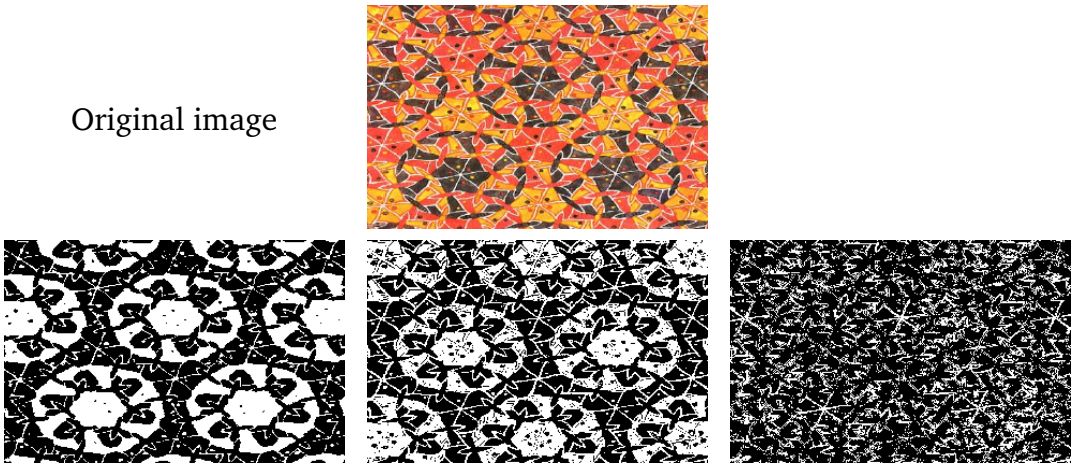


Figure 4.12: Result of MVGen on an Escher image

the liver disorders data, we clearly find different groupings in these views (small rand index), and the views do not correspond to the full-dimensional space, i.e. for each view some dimensions are irrelevant. For liver, we observe disjoint views. The first view clearly describes the relation between alcohol consumption and the mean corpuscular volume, while the second view represents the weaker indicators. On the diabetes data, the detected views match well to some factors causing diabetes of type 1 or type 2 (adult-onset diabetes; also caused by high blood glucose levels during pregnancy). Here, a further interesting observation can be made: Besides finding dissimilar groupings in subspace projections, we now also get slightly overlapping views. E.g., the dimension 'body mass index' is relevant for both clusters in view 1 and for a single cluster in view 2. This result also confirms our hypothesis that the clusters of the *same* view may slightly differ in their relevant dimensions.

Overall, our experiments show that MVGen successfully detects the multi-view clustering structure on a variety of real world data sets.

4.5 Conclusion

In this chapter we introduced the MVGen approach to learn the clustering structure of data containing multiple overlapping views. To this end, we proposed a novel generative model: it represents the different views of the data by multi-

Liver Disorders Data (rand index between views: 0.25)	
V1	mean corpuscular volume, number of half-pint equivalents of alcoholic beverages drunk per day
V2	alkaline phosphotase, alamine aminotransferase, aspartate aminotransferase, gamma-glutamyl transpeptidase
Diabetes Data (rand index between views: 0.51)	
V1	body mass index, diabetes pedigree function, triceps skin fold thickness, 2-hour serum insulin; (for one cluster: plasma glucose concentration)
V2	age, diastolic blood pressure, # of times pregnant, plasma glucose concentration; (for one cluster: body mass index)

Table 4.1: Subspace views on liver and diabetes

ple mixture models located in different subspace projections and it, thus, allows each object to participate to various clusters. Since for each view a partitioning of the objects is performed, we are able to describe multiple groupings and at the same time prevent redundant information. In our method, we use Bayesian model selection to decide which sets of dimensions are relevant for the clusters and views. Since each cluster is detected in an individual subspace projection, the problems of full-space clustering are avoided. For an efficient learning, we exploit the principle of iterated conditional modes in our MVGen approach, and we derived the required update equations. The comparison of MVGen with competing approaches demonstrated the strengths of detecting views in multiple subspace projections. Our MVGen approach was able to discover multiple clustering views for various real world data sets. Especially the explicit modeling of the views' relevant subspaces has proven to be very valuable for interpreting the final clustering results.

Chapter 5

Subspace Correlation Clustering

In the previous chapters of this thesis we have shown that analyzing subspace projections and avoiding redundant information leads to meaningful clustering results. With our methods, we avoid analyzing the full space in which the clustering structure may be obfuscated by irrelevant dimensions and multiple views. While our previously described methods focus on detecting dense areas in the data space, i.e. groups of objects are characterized by similar attribute values, we now consider a more complex pattern: correlation. For this type of pattern, a group of objects is regarded as similar if their attribute values depend on each other in a similar way. As we will demonstrate, the observations that originate the need to consider subspace projections for traditional clustering also apply for the task of correlation analysis.

5.1 Motivation

The goal of correlation clustering[1] is to identify groups of objects that exhibit dependencies between the features of the dataset. Considering correlations, the attribute values in one dimension depend on the attribute values of other dimensions. In contrast to clustering, however, the objects do not have to be closely located together but should describe the same regression, e.g., they are located near the same line or plane. Knowledge about the dependencies of attributes

[1]Not to be confused with the term as used in the machine learning community, which refers to the task of determining a partitioning of the data that correlates to a pairwise similarity function f learned from past data [BBC04].

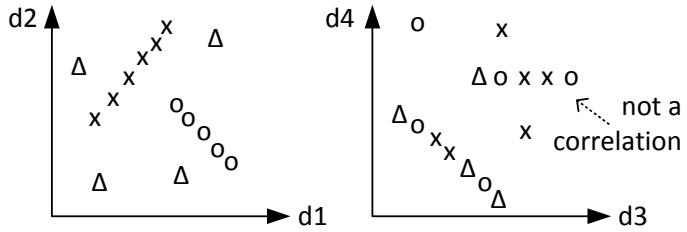


Figure 5.1: 4-d database with 15 objects and 3 subspace correlation clusters

provides a motive to reason about causalities and is advantageous for, e.g., trend analysis, decision strategies, and prediction.

Mistakenly, in the literature, correlation clustering is characterized as generalized subspace clustering [KKZ09]. Contrarily, we show in the following how traditional correlation clustering falls short of the fundamental observations and assumptions which originated the research field of subspace clustering and that the actual transfer of the correlation principles towards subspace clustering still needs to be done.

Subspace clustering [PHL04, KKZ09] identifies clusters in any (iso-oriented) subspace projection of the data. In consequence, each group of objects can have an individual set of relevant dimensions. Similarly, correlation clustering assumes that strong correlations can only be observed for a subset of objects and a subset of attributes. The assumption of each cluster having its own subset of relevant attributes led *subspace clustering* to realize that an object can easily belong to *multiple clusters* when considering different views, i.e., attribute subsets of the data (cf. Chapter 3 and 4). This aspect, however, is not considered by the existing correlation clustering methods, which require the clusters to be (almost) disjoint with respect to their objects.

By considering the 2-dimensional subspace $\{d1, d2\}$ in Figure 5.1, two different (local) correlations can be detected: The objects indicated by a cross are positively correlated on a line, while the objects indicated by a circle are negatively correlated on a different line[2]. Considering the subspace $\{d3, d4\}$, a different correlation supported by a different set of objects can be detected. Thus, objects may contribute to several correlations due to different subspace projections.

A second observation is the following: Given a multitude of overlapping clusters in different subspaces and a set of locally irrelevant dimensions, an immedi-

[2]Of course our paradigm is not restricted to lines but correlations with higher degrees of freedom may be detected.

ate consequence is that the clustering structure might be totally obscured in the full-space [BGRS99]. That is, due to the decreasing discrimination power of distances, we cannot find a clustering solution that stands out in the data; almost any grouping looks alike when, e.g., comparing their compactness in the full-space. By analyzing subspace projections, as, e.g., done in our RESCU, OSCLU and MVGen method, the discrimination power of distances and hence between different clustering solutions is increased, and clearly outstanding clusters can be detected. In existing correlation clustering methods, however, such an obfuscation of the clustering structure in the full-space is not taken into considerations.

Why do we highlight this problem? One might argue that *given* an appropriate set of objects, correlation clustering can easily determine the subset of correlated dimensions and excludes the irrelevant dimensions by a post-processing. The crux, however, is to *find* the 'appropriate' set of objects in the first place. If patterns are hidden in subspace projections of the data as, e.g., illustrated in Fig. 5.1, this is an almost impossible task for approaches operating in the full-space. Since correlation clustering analyses the full-space, noisy dimensions and overlapping clusters will obfuscate the local correlations of objects; none of the possible groupings stands out in the full dimensional data space (similar to traditional full-space clustering). In Fig. 5.1 it is very unlikely to find a group of objects in the full-space that corresponds to a strong correlation in a subspace. If, however, already wrong sets of objects have been clustered, determining the true subset of correlated dimensions in a post-processing is almost impossible. Thus, correlation clustering will not just miss some clusters due to its partitioning, but the resulting clusters are highly questionable as for the traditional full-space clustering approaches.

A novel mining paradigm. In this chapter, we show that the observations made for (subspace) clustering also have to be transferred to the task of correlation analysis and we introduce the *novel paradigm of subspace correlation clustering*: First, correlations between dimensions can be restricted to subspace projections, i.e., not the whole set of dimensions is correlated. And second, correlations can be hidden in local and possibly overlapping patterns, i.e., only subsets of the objects may be correlated, and objects may contribute to several correlations.

With our paradigm we detect locally correlated dimensions in subspace projections. If we considered the full-dimensional data space in Fig. 5.1, no strong correlation between the dimensions could be detected. Consequently, we focus on subspace projections of the data. Overall, subspace correlation clustering derives

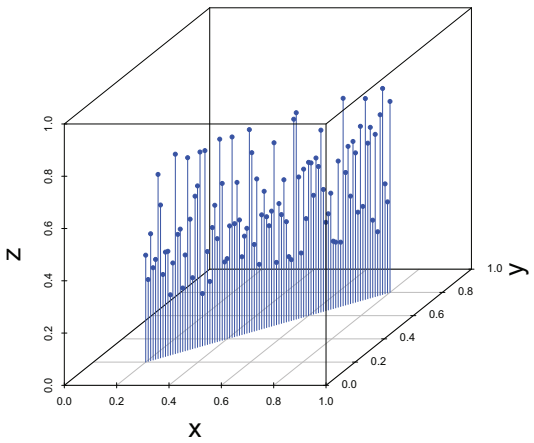


Figure 5.2: Valid correlation in subspace $\{x, y\}$, invalid in $\{x, y, z\}$

more meaningful and more precise local patterns, and it unfolds the potential of correlation clustering also for higher dimensional datasets.

Furthermore, in our paradigm we aim at reporting only clusters whose dimensions are (strongly) correlated. In Fig. 5.2, for example, any existing correlation clustering approach would identify the plane as a cluster; though, the attribute values in z do not depend on the values in x and y. Highly misleading information can be derived from such results: the interpretation that all three dimensions are correlated on a plane is wrong! Thus, in our paradigm we aim at avoiding such misleading clusters. Instead of detecting the plane in the 3d-subspace, our gaol is to detect the line in subspace $\{x, y\}$ since these two dimensions are strongly correlated. Similarly, in Fig. 5.1 the horizontal line in subspace $\{d3, d4\}$ does not correspond to a correlation. The attribute values do not depend on each other. Our method will just determine the other three lines in subspace $\{d3, d4\}$ and $\{d1, d2\}$. Since we report only the truly correlated dimensions to the user, which exactly represent the different subspaces, misinterpretations of the clustering result are prevented.

Multiple views and redundancy. By analyzing subspace projections, objects can naturally be included in multiple patterns (see Fig. 5.1). While allowing each object to contribute to several correlations is beneficial for detecting multiple and alternative patterns in the data, we have to cope with the challenge of redundancy. As known from the previous chapters, analyzing arbitrary subspace projections and simply reporting any detected cluster results in highly redundant information. This observation also holds for the task of correlation analysis. For example, a line detected in subspace S is also a line in any of its projections $S' \subseteq S$ (with

$|S'| \geq 2$). Including these projections in the final result should be avoided as they represent similar correlations and are not beneficial for the user. Even worse, some correlations just exist due to other patterns, hence, representing misleading (and redundant) information. Thus, to ensure interpretable results for our novel paradigm, we have to develop a redundancy model handling these scenarios.

Overall, the contributions of this chapter for the novel paradigm of subspace correlation clustering are:

- we analyze *subspace projections* to find *correlated dimensions* supported by a *subset of objects*, i.e., we perform local pattern subspace correlation analysis
- we allow multiple overlapping clusters, i.e., each object can contribute to several correlations due to different subspace projections
- we avoid redundancy in the clustering result, i.e., we remove clusters representing similar and misleading correlation information

5.2 Comparison with Related Work

In the following we review paradigms that are related to the topic of subspace correlation clustering. The general clustering objective is to group objects based on their mutual similarity. For traditional clustering, the similarity of vector data is determined based on all attributes of the feature space, and the goal usually is to find clusters that excel by a high compactness or density in the original feature space.

Subspace Clustering: The key to the paradigm of subspace clustering [KKZ09] is the observation that irrelevant dimensions and multiple possible groupings obscure the clustering structure in the full-space due to the 'curse of dimensionality' [BGRS99]. Since no grouping stands out in the data, traditional full-space clustering often leads to questionable clustering results. Consequentially, to increase the discrimination power, subspace clustering techniques aim at identifying clusters in different attribute subsets of the data. Thus, depending on the selected subspace different clusters are possible and objects may occur in a multitude of clusters. The main challenge for subspace clustering is the avoidance of redundancy in the clustering result as presented in Chapters 2-4 of this thesis. Obviously, existing subspace clustering methods are not designed to detect groups of objects describing correlations among the dimensions.

Projected Clustering: As discussed in Chapter 2.1, projected clustering is related to subspace clustering but does not allow multiple overlapping clusters, i.e. projected clustering performs (almost) an partitioning of the objects. Consequently, projected clustering misses many of the hidden clusters, or the resulting solution is of low quality since it tears multiple clusters apart to fit one partitioning. Projected clustering methods are also not suited for detecting correlations in the data.

Correlation Clustering: Methods from this paradigm aim at identifying object groups describing correlations between different features. Since the dimensions of a cluster are not restricted to subsets of the original attributes but correspond to arbitrarily oriented subspaces, correlation clustering is often denoted as generalized *subspace* clustering [KKZ09]. This, however, is (at least) inaccurate! Existing correlation clustering methods are rather generalized *projected* clustering methods and accordingly have the same limitations. They are not able to find multiple overlapping subspace correlation clusters, since they are limited to find only disjoint or, in the case of [AR10], nearly disjoint clusters. Furthermore, existing methods ignore the obfuscation of the clustering structure in the full-space provoked by highly overlapping clusters in different subspaces.

For completeness, we review existing correlation clustering methods and discuss further shortcomings. We will restrict the considerations to linear correlation clustering in the following. The basic technique utilized by most approaches is Principal Component Analysis (PCA). Agglomerative methods (ORCLUS [AY00], 4C [BKKZ04], COPAC [ABK$^+$07b], HiCo [ABKZ06], ERiC [ABK$^+$07a]) assume that local neighborhoods are in line with a global trend and perform PCA on those neighborhoods, which, however, are strongly influenced by the similarity in the full-space. Therefore these techniques are not appropriate for noise or multiple views in the data as both negatively influence the choice of the neighborhood. To avoid the problem of defining a proper local neighborhood, divisive methods were proposed [LHSY07], [AR10], which, however, usually require the number of clusters to be specified beforehand. Instead of PCA the approach of [ABD$^+$08] utilizes the Hough transform to detect correlations within the attributes. Furthermore, as shown in the introduction, the previous approaches may include uncorrelated dimensions (e.g., the horizontal line in Fig. 5.1) that hinder the interpretation of the clustering result. In [HH07] uncorrelated dimensions are avoided but they are restricted to find disjoint lines.

The paradigm of pattern-based clustering [YWWY02, KKZ09], such as biclustering or co-clustering, is related to correlation clustering in that it clusters objects

that exhibit a similar trend in a subset of attributes. However, it is limited to one-dimensional positive correlations, i.e., all attributes in the subset are correlated pairwise.

Overall, none of the existing correlation clustering methods is able to find multiple overlapping subspace correlation clusters as it is possible with our method.

Subspace correlation analysis: Besides clustering, the paradigm of subspace correlation analysis has to be mentioned. Approaches as CARE [ZPW08a] and REDUS [ZPW08b] try to identify subsets of attributes in which the *majority* of data entries exhibit a correlation. This differs from clustering in that it does not analyze *several local subsets* of objects. Therefore these methods are restricted to finding only a single correlation per subspace and the datasets have to contain a small degree of noise concerning objects.

5.3 Subspace Correlation Clustering

Generally speaking, a subspace correlation cluster is a set of objects $O \subseteq DB$ that exhibits a correlation in a set of dimensions $S \subseteq Dim$. We call this set of attributes the subspace of the cluster. In Fig. 5.2 for example, the whole set of objects is correlated in the 2-dimensional subspace $\{x, y\}$; the objects form a line. Considering the 3-dimensional space, however, we do not have a valid subspace correlation cluster since the attribute z is not correlated to the other ones. Correlation is a measure of the effect of independent variables on a dependent variable [CCWA02]. As shown in Fig. 5.2, however, we cannot draw any conclusion about the attribute values in dimension z given the attribute values in the remaining dimensions. Thus, the plane does not represent a valid subspace correlation cluster.

Considering Fig. 5.3 the illustrated plane consisting of all objects in the 3-dimensional space, however, is a valid correlation. Given the attribute values of an object in dimension x and y, the attribute value in dimension z depends on x, y. An object in this correlation is described by *two independent variables*. If we just consider a subset of the objects, we are even able to find a line in the 3-dimensional subspace, i.e., given the attribute value in dimension x, for example, we observe a dependency of the attribute values in dimension y and z. For a line we have *one independent variable*, i.e., one degree of freedom.

As shown in the examples, for subspace correlation clusters we have two different types of dimensionalities: First, the *subspace dimensionality*: for example,

the 3-dimensional subspace of $\{x, y, z\}$. Second, the *cluster dimensionality*: Within each subspace, each cluster/correlation has its own intrinsic dimensionality. E.g., a plane corresponds to a 2-dimensional cluster (two degrees of freedom) while a line is a 1-dimensional cluster. Thus, in general, a subspace correlation cluster is defined by a triple (O, S, λ) with a set of objects $O \subseteq DB$, a set of correlated dimensions $S \subseteq Dim$, and the cluster's dimensionality λ, representing the degrees of freedom. In the following, we formalize the definition of such subspace correlation clusters.

5.3.1 Cluster definition

The basic idea to describe correlations in the data is to consider the data's principal components. We adapt the notions of [ABK+06] to define our clusters. Since we design our model to detect linear correlations, we will use the term 'correlation' in the place of 'linear correlation'.

Definition 5.1 *Basic notions*
We assume a database $DB \subseteq \mathbb{R}^d$ of d-dimensional objects is given. $Dim = \{1, \ldots, d\}$ is the set of dimensions. With $o|_S$ we denote the projection of an object $o \in DB$ to the subspace $S \subseteq Dim$. Accordingly, $O|_S$ refers to the projection of a set of objects $O \subseteq DB$. The (sample) covariance matrix for $O|_S$, i.e., for a set of objects $O \subseteq DB$ in a subset of dimensions $S \subseteq Dim$, is denoted by $\Sigma_{O,S} \in \mathbb{R}^{|S|} \times \mathbb{R}^{|S|}$. The eigendecomposition of $\Sigma_{O,S}$ is $\Sigma_{O,S} = V_{O,S} \cdot E_{O,S} \cdot V_{O,S}^T$. The eigenvalue matrix $E_{O,S}$ is a diagonal matrix storing the $|S|$ eigenvalues in decreasing order, i.e., $E_{O,S} = diag(e_1, \ldots, e_{|S|})$ with $e_1 > e_2 > \ldots > e_{|S|}$. The eigenvector matrix $V_{O,S}$ is an orthonormal matrix storing the unit eigenvectors v_i corresponding to e_i. Therefore, the i-th principal component is $e_i \cdot v_i$.

Given a set of objects O projected to the subspace S, i.e., $O|_S$, the minimum number of principal components needed to retain a significant level α of the data's variance corresponds to the cluster's intrinsic dimensionality. Considering the line (defined by a subset of the objects) in the 3-dimensional subspace in Fig. 5.3, we just need one principal component to describe most of the variance. This is indicated by one large eigenvalue and two small ones. In contrast, for the plane we need two principal components. Thus, by using the eigenvalues of the covariance matrix, we are able to determine the cluster dimensionality.

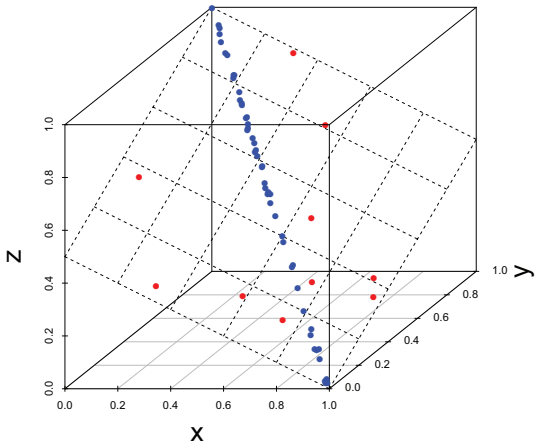


Figure 5.3: Redundant plane (induced by the non-redundant line)

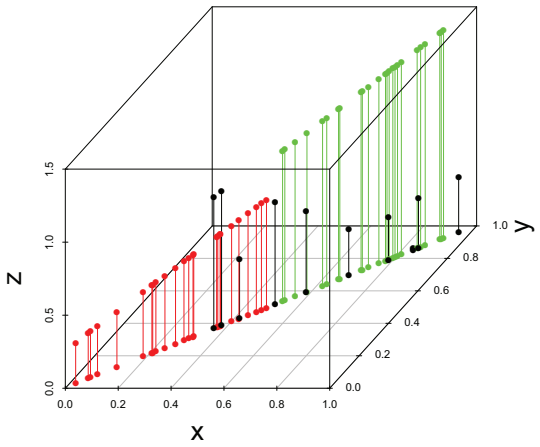


Figure 5.4: Redundancy (line in subspace $\{x, y\}$) due to collinearity

Definition 5.2 *Cluster dimensionality*
The cluster dimensionality of the (projected) set of objects $O|_S$ w.r.t. a significance level α is

$$\lambda(O,S) = \min\{k \in \mathbb{N}^+ \mid \frac{\sum_{i=1}^{k} e_i}{\sum_{i=1}^{|S|} e_i} \geq \alpha\}$$

using the entries e_i of the eigenvalue matrix $E_{O,S}$.

Considering again the line in the 3-dimensional subspace of Fig. 5.3, we get the eigenvalues $e_1 \approx 10$ and $e_2 \approx e_3 \approx 0.01$. Thus, by selecting, for example, $\alpha = 0.9$, the cluster dimensionality corresponds to 1, as expected for a line. By increasing the value of α also the clusters' dimensionalities increase since a larger fraction of the variance needs to be explained.

Since real world data is rarely perfect but contains errors and noise, we cannot expect to observe a set of objects that perfectly fits a line or another linear regression model. Thus, the objects slightly deviate from the perfect model, e.g. in Fig. 5.3 we observe a small deviation of the objects from the line. We can measure the strength of a correlation by the degree of variation *not* explained by the regression model: Since the regression model corresponds to the hyperplane spanned by the first $\lambda(O,S)$ principal components with the largest eigenvalues (strong principal components), the distance along the last $|S| - \lambda(O,S)$ principal components with the smallest eigenvalues (weak principal components) determines the strength of a correlation. Intuitively, this corresponds to the (perpendicular) distance between an object and, e.g., the perfect line. The smaller these distances, the stronger the correlation.

Formally, the correlation distance can be computed by determining the Euclidean distance after projecting the objects and the cluster mean onto the weak principal components (note that the objects are firstly projected to the subspace S, that is, the weak components are linear combinations of the dimensions in S):

Definition 5.3 *Subspace correlation distance*
The subspace correlation distance of an object $p \in DB$ to a correlation defined by a set of objects O in subspace S is given by

$$scdist(p, O, S) = \sqrt{(p|_S - \mu|_S)^T \cdot V_{O,S} \cdot \hat{E} \cdot V_{O,S}^T \cdot (p|_S - \mu|_S)}$$

with μ is the mean vector of the objects in O, $V_{O,S}$ is the eigenvector matrix, and $\hat{E}$ is a diagonal matrix where the first $\lambda(O, S)$ entries are 0 and the remaining $|S| - \lambda(O, S)$ entries are 1.

Thus, we obtain a strong correlation if each object of the cluster has a small subspace correlation distance. Using these basic ideas, we first introduce our novel cluster definition and afterwards we highlight the important characteristics.

Definition 5.4 *Subspace correlation cluster*
A subspace correlation cluster $C = (O, S, \lambda)$ is a set of objects $O \subseteq DB$, a set of dimensions $S \subseteq Dim$, and its cluster dimensionality $\lambda = \lambda(O, S)$, such that

1. *the cluster is sufficiently large, i.e., $|O| \geq minSize$*
2. *the subspace correlation distance is small for each object in the cluster, i.e., $\forall o \in O : scdist(o, O, S) \leq \varepsilon$*
3. *the set of objects is maximal: any object not in the cluster has a larger subspace correlation distance, i.e., $\forall p \in DB \backslash O : scdist(p, O, S) > \varepsilon$*
4. *the cluster dimensionality is smaller than the subspace dimensionality, i.e., $\lambda < |S|$*
5. *uncorrelated dimensions are not included, i.e., $\forall d \in S : \lambda(O, S \backslash \{d\}) = \lambda(O, S)$.*

The first property ensures that each cluster has sufficiently large support since, e.g., a line composed by only two points is not interesting. By the second property, we ensure that each cluster exhibits a strong correlation; we do not include objects with large subspace correlation distances. Accordingly, we want to include any object with a small distance, i.e., the cluster should be as large as possible (third property).

The last two properties are the most important ones: With Prop. 4 the case $\lambda = |S|$ is prohibited ($\lambda > |S|$ is not possible by definition). For $\lambda = |S|$, the cluster dimensionality and the subspace dimensionality would be identical. This is not meaningful since each dimension in the current subspace would be an independent variable. Thus, none of the attribute values depends on the others, which does not represent a correlation.

Property 5 is even more important to avoid "meaningless" correlations (and is one large advantage in contrast to traditional correlation clustering). As shown in Fig. 5.2, the plane is not a valid correlation; dimension z is not correlated to the other ones. This, however, cannot be detected by considering just the eigenvalues. The cluster dimensionality is 2 as for any other plane. Thus, the first 4 properties of the previous definition hold. However, we can detect this invalid correlation by the following principle: Lets assume we have detected a set of objects with dimensionality λ' in subspace $S\backslash\{d\}$. By adding a further dimension, i.e., to get the subspace S, the novel cluster dimensionality obviously has to be either $\lambda'+1$ or λ'. The first case occurs if we add an uncorrelated/noisy dimension to $S\backslash\{d\}$, e.g., a line in a 2-d subspace becomes a plane in a 3-d subspace (cf. Fig. 5.2). The latter case occurs if we add a correlated dimension, e.g., a line still is a line (Fig. 5.3). Thus, Property 5 checks for each lower-dimensional subspace $S\backslash\{d\}$ whether the cluster dimensionality remains stable. Otherwise ($\lambda(O, S\backslash\{d\}) < \lambda(O, S)$) at least one dimension d would be uncorrelated and the cluster is not valid. (Note that an uncorrelated dimension can also be detected by checking if a principal component of the cluster is nearly parallel to the dimension's axis.)

Overall, our cluster definition enables us to detect groups of objects O that are correlated in the set of dimension S with a specific degree of freedom λ by simultaneously excluding uncorrelated dimensions. Note that the *smaller* λ the more interesting is a cluster; a line is better than a plane. Contrarily, the *larger* S the more interesting is a cluster; a 3-d subspace is better than a 2-d subspace.

5.3.2 Clustering definition

Based on Definition 5.4 we are able to determine all valid clusters, which are allowed to group diverse object sets in diverse subspaces. Thus, we allow overlapping clusters in general and we are able to detect multiple different correlations per object, e.g., due to different attribute subsets. We are not limited to disjoint clusters as previous methods. However, simply using the set All of all valid sub-

space correlation clusters according to Definition 5.4 would lead to an overwhelming result size, containing highly redundant information. This redundancy occurs due to two reasons:

Collinearity: When one dimension $d \in S$ is highly correlated with some other dimension x, the remaining dimensions $S \backslash \{d\}$ will be correlated to x, too. This phenomenon is called (multi)collinearity [FW06]. When this happens, we can derive many clusters describing the same information, e.g., if $\{d1, d2\}$, $\{d2, d3\}$ and $\{d3, d4\}$ are correlated, we directly can infer that $\{d1, d3\}$, $\{d1, d4\}$, ... are correlated as well. Instead of reporting several collinear correlations in low dimensional subspaces, we represent them by just one cluster in a subspace S of higher cardinality (note: the cluster dimensionality λ remains unchanged). This way, we avoid redundancy.
To handle collinearity, we have to ensure that a correlation cluster in subspace S is not simply a projection of another cluster in a superspace $S' \supset S$. Fig. 5.4 illustrates this problem. The correlation cluster in subspace $\{x, y\}$ is a projection of two clusters in subspace $\{x, y, z\}$ (the green and red clusters) and a few noise objects (black dots). Such a cluster does not provide much additional insight and therefore should be excluded from the final result.

Induced clusters: With a few extra points, a correlation cluster can induce a cluster of higher dimensionality (e.g., a line induces a plane). In Fig. 5.3, the 1-dimensional correlation cluster (blue line) induces the 2-dimensional correlation cluster (plane) which has only a few extra objects. Induced clusters can be misleading as they mainly take credit for another cluster's objects. In this example, the plane is mainly supported by the objects of the line. If we took the line's objects out of consideration, the plane would have too little support to be a valid correlation cluster. The plane is just valid because of the line but not by itself. Thus, we should exclude induced clusters from the final result.

The 'potentially redundant' rule. The main question is, which clusters could lead to a redundancy of other clusters either due to collinearity or due to induction. The following definition clearly states this:

Definition 5.5 *'Potentially redundant' rule*
A subspace correlation cluster $C = (O, S, \lambda)$ is potentially redundant to a cluster $C' = (O', S', \lambda')$, for short $C \prec_{red}^{pot} C'$, iff

$$\lambda - \lambda' \geq |S \backslash S'| \vee \exists S^* \subset S : |S^*| = \lambda + 1 \wedge \lambda - \lambda' > |S^* \backslash S'|$$

We continue by discussing why the above rule covers all clusters C' to which C is potentially redundant to. We distinguish three cases:

1) $\lambda < \lambda'$ (the cluster dimensionality of C is lower (better) than the ones of C'):
In this case, the rule is always evaluated to $false$ as the left sides of both inequalities are negative while their right sides are at least 0. This means a lower dimensional cluster (e.g., a line) is never redundant with respect to a higher dimensional cluster (e.g., a plane). This is desirable because high-d clusters cannot induce low-d ones.

2) $\lambda = \lambda'$ (both clusters have the same dimensionality):
In this case, the rule will be true if and only if $S' \supseteq S$. Otherwise the right side of the first inequality will be greater than 0, and the second part of the rule is violated in any case since a strict $>$ is required. This means a cluster in a higher-dimensional subspace is more important than its projections. As we can derive clusters in lower-dimensional subspaces by simply projecting the cluster of a higher-dimensional subspace, clusters in lower-dimensional subspaces need not to be presented explicitly (cf. collinearity).

3) $\lambda > \lambda'$ (the cluster dimensionality of C is higher (worse) than the ones of C'):
This case is the most complex one and it corresponds to induced correlation clusters, e.g., C is a plane while C' is a line. As before, the rule is true for the case $S' \supseteq S$. Though, it can even hold for $S' \not\subseteq S$.

Let us consider an example: C is a 4-d cluster in subspace $S = \{1, 2, 3, 4, 5\}$ and C' is a 1d-cluster (line) in subspace S'=$\{1, 2, 3, 8, 9\}$. How does C' look like in S? If we project C' to $\{1, 2, 3\}$=$S \cap S'$, it has to be still a line. By now adding two further dimensions, e.g., $\{4, 5\}$ to reach S, the dimensionality of the 1d-cluster increases at most by two (if both dimensions are uncorrelated). Thus, in the worst case the dimensionality of C' in S is $1+2=3$, which is better than the 4d-cluster C. Hence C' could potentially induce C. In general, this holds if $\lambda' + |S| - |S' \cap S| < \lambda \Leftrightarrow |S \setminus S'| < \lambda - \lambda'$.

But there is more: Now assume C to be a 4-d cluster in the subspace $S = \{1, 2, \ldots, 7\}$. Obviously, the line C' does look like a 5-d cluster in S. It cannot induce the cluster C. However, in our model the cluster C represents all of its collinear projections, e.g., also the 4-d cluster in subspace $\{1, 2, 3, 4, 5\} = S^* \subset S$. As discussed above, the cluster in this subspace may be induced by C' and hence could be redundant. Consequently, if the projection S^* is redundant then this information should not be contained in the 'parent' cluster. We also have to denote

C as potentially redundant to C'. In general, the equation $\lambda - \lambda' > |S^* \backslash S'|$ has to be checked for any subset $S^* \subset S$ with $|S^*| = \lambda + 1$ since these correspond to the lowest dimensional subspaces represented by collinear information of C.

Overall model. Based on the 'potentially redundant' rule, we are now able to define the overall redundancy of a cluster. As discussed and illustrated in Fig. 5.3, the plane should be discarded as redundant because its support is too small after removing the line. Similarly, in Fig. 5.4, the line in subspace $\{x, y\}$ should be discarded, since the support is mainly due to the two lines in subspace $\{x, y, z\}$. Thus, we define a cluster C as redundant w.r.t. a set of other clusters $Result$ if C's support is too small after removing all objects that are already grouped in clusters $C' \in Result$ to which C is potentially redundant.

Definition 5.6 *Redundancy of clusters*
A subspace correlation cluster $C = (O, S, \lambda)$ is redundant to a set of clusters $Result$, for short $C \prec_{red} Result$, iff

$$|O \setminus \bigcup_{\substack{C' \in Result \setminus \{C\} \\ C \prec_{red}^{pot} C' \\ C' = (O', S', \lambda')}} O'| < minSize$$

Finally, we are able to define the overall subspace correlation clustering. The final clustering should be redundancy free, i.e., it should not contain induced clusters or clusters present due to collinearity. Moreover, it should be maximal, i.e., should contain as many clusters as possible without introducing redundancy. Based on these two principles, we define the overall clustering model as follows:

Definition 5.7 *Subspace correlation clustering*
Given the set All of all valid subspace correlation clusters, a subspace correlation clustering $Result \subseteq All$ fulfills the following conditions:

- *redundancy-free:* $\forall C \in Result : C \not\prec_{red} Result$
- *maximality:* $\forall D \in All \backslash Result : Result \cup \{D\}$ *is not redundancy-free*

Our novel clustering model enables the detection of correlations in subspace projections of the data, it allows objects to contribute to multiple, overlapping correlations, and simultaneously prevents redundant information in the final output.

5.3.3 Discussion of parameters

Our subspace correlation clustering model can be adapted based on three parameters, which were already introduced by the method of COPAC [ABK+07b]: α is used to determine the cluster dimensionality of a set of objects O, i.e. it controls the degrees of freedom of the corresponding correlation. At least α percent of O's variance has to be explained by the calculated correlation. According to [ABK+07b] one typically selects $\alpha \in [0.8, 0.9]$. In this case, the correlation has to capture most of the information reflected in the set of objects O.

The parameter ε ensures that all but only similar objects (w.r.t. the correlation distance) are included in the cluster. By choosing $\varepsilon = 0$, the objects of a cluster have to form a perfect correlation; the object values exactly fit the regression model. By increasing the value of ε, larger deviations from the regression model are permitted. Thus, the method is more robust to errors or noise in the data. Intuitively, ε corresponds to the maximal error the user is willing to accept for each correlation. Similar to density based clustering [EKSX96], all objects with larger distances are excluded from the cluster.

The last parameter $minSize$ simply restricts the minimal cardinality of each cluster. Thus, meaningless correlations as, e.g., two objects forming a line are prevented. By increasing $minSize$, the detected clusters reflect stronger correlations in the data since they are supported by a larger set of objects. Obviously, the number of clusters decreases while $minSize$ is increased.

5.4 The SSCC Algorithm

In the following section we briefly introduce our algorithm SSCC (SubSpace Correlation Clustering), which determines a clustering result according to Definition 5.7. We develop an efficient algorithm computing an approximate solution since, as known from traditional subspace clustering, the number of cluster candidates is exponential in the number of objects and the number of dimensions. Furthermore, generating all cluster candidates in a first step and selecting the final clustering afterwards is highly inefficient since most of the clusters will be rejected as redundant anyway. Thus, our SSCC avoids the bottleneck of generating all possible clusters by trying to directly generate only non-redundant clusters.

The general processing of SSCC is shown in Algo. 5.1. We first describe the gen-

Algorithm 5.1: The SSCC algorithm

```
1  Result = ∅ // current result containing non-red. clusters
2  for λ from 1 to d − 1 do // low-dim. clusters first
3      λClusters = ∅
4      for S ⊆ Dim with |S| = λ + 1 do
5          NonPruned_S=PruneObjects(DB,S,Result,λClusters) // cf. Section 5.4.3
6          if |NonPruned_S| < minSize then continue;
7          Tmp=FindλClusters(NonPruned_S, S, λ) // cf. Section 5.4.1
8          λClusters=MergeClusters(Tmp, λClusters) // cf. Section 5.4.2
9          // λClusters may now contain clusters with
10         // subspace cardinality |S_C| > λ+1
11     for x from d to λ + 1 do // high-d subspaces first
12         for C ∈ λClusters with |S_C| = x do
13             if C is non-redundant to Result then // cf. Def. 5.6
14                 Result = Result ∪ {C}
```

eral idea of our approach and discuss details in the subsequent paragraphs. Three major principles are used to avoid generating redundant clusters: *(1)* Based on the redundancy definition, a λ-dim. cluster can only be redundant to clusters with dimensionality $\lambda' \leq \lambda$. Thus, we can first mine all (non-redundant) low-dimensional clusters (e.g., lines) before mining higher-dimensional ones (e.g. planes). This is shown in line 2 of the algorithm. *(2)* For a non-redundant λ-dim. cluster in subspace S, its support has to be high enough *after* removing the objects contained in clusters that make him potentially redundant (cf. Def. 5.5). We use this idea for pruning objects: Since we already know the non-redundant λ'-dim. clusters with $\lambda' \leq \lambda$ in similar subspaces, we remove all objects of the database (in this subspace) that are already clustered (line 5). Thus, the set of objects to be considered is dramatically reduced (or even too small; line 6) and finding clusters is more efficient. *(3)* We exploit the fact of collinearity to avoid analyzing the exponential number of possible subspaces. Each λ-dim. cluster in subspace S is also a λ-dim. cluster in any subspace $S' \subseteq S$ with $|S'| = \lambda+1$. Thus, we only mine λ-dim. clusters in subspaces with cardinality $\lambda+1$ (cf. line 4, 7) and we *merge* these clusters to obtain clusters with higher subspace cardinality (line 8). E.g., if we have already found similar clusters in subspace $\{d1, d2\}$ and $\{d1, d3\}$, we merge them to $\{d1, d2, d3\}$. If the cluster is still valid here, we do not have to analyze $\{d2, d3\}$ since in our model the collinearly correlated dimensions are represented by a single cluster.

Based on these strategies, the lines 3-10 of Algo. 5.1 efficiently generate a set of λ-dim. clusters (located in subspaces of arbitrarily high cardinality) that is non-redundant with high probability. However, the previous steps do not perfectly guarantee a non-redundant result. Thus, in lines 11-14 we finally ensure the absence of redundancy: For this, we can use an efficient incremental approach since based on Definition 5.5 we can process the clusters of highest *subspace* cardinality first (line 11,12) and we just have to compare these against the current result set (line 13).

5.4.1 Finding λ-dim. clusters in a $(\lambda + 1)$-dim. subspace

As described above, we just have to mine the λ-dim. clusters in $(\lambda + 1)$-dim. subspaces. To achieve this, we apply the method of COPAC [ABK+07b] on the current subspace projection S; however, with two important differences: 1) The basic idea of COPAC is to compute for each object o its local correlation dimensionality λ_o, i.e. the dimensionality of the correlation if only the objects in o's neighborhood are considered. Afterwards, COPAC partitions the objects according to their local correlation dimensionality, and finds clusters in *any* of these partitions. Since we are just interested in λ-dimensional clusters, we do not have to analyze all partitions but just a single partition; the one corresponding to λ. Thus, we only have to find clusters in the partition that contains the objects o having a local correlation dimensionality of $\lambda_o = \lambda$. This is far more efficient than analyzing all objects. Note that we still find clusters of higher dimensionality due to our overall processing. 2) As mentioned, existing methods may determine clusters containing uncorrelated dimensions. In contrast, our approach avoids these clusters, i.e., clusters containing principal components that are nearly parallel to any axis of the current subspace are rejected. Overall, based on our modification of COPAC, we efficiently generate the desired set of λ-dim. clusters for a given subspace S.

5.4.2 Merging clusters to higher-dimensional subspaces

With our merging principle we avoid to analyze any possible subspace projection. Given the set Tmp of newly detected λ-dim. clusters (cf. line 7), we try to merge these with the already known $\lambda Clusters$ to reach subspaces of higher cardinality. A pseudo-code for this subroutine is given in Algo. 5.2. For each cluster $C = (O, S, \lambda) \in Tmp$ we first determine those $C_i = (O_i, S_i, \lambda) \in \lambda Clusters$ that fulfill

Algorithm 5.2: Generate high-dimensional subspaces by merging

```
   function: MergeClusters(.,.)
1  input: new clusters Tmp, already known λClusters
2  while Tmp ≠ ∅ do
3      select C = (O, S, λ) ∈ Tmp
4      PotColl = {C_i ∈ λClusters | |S ∩ S_i| ≥ λ ∧ S_i ⊈ S}
5      CIsRedundant = false
6      for C_i ∈ PotColl do
7          generate C_merge = (O ∩ O_i, S ∪ S_i, λ)
8          if C_merge is valid cluster then
9              Tmp = Tmp ∪ {C_merge}
10             if C_i is redundant to C_merge then
11                 λClusters = λClusters\{C_i}
12             if C is redundant to C_merge then
13                 CIsRedundant = true
14     Tmp = Tmp\{C}
15     if CIsRedundant == false then
16         λClusters = λClusters ∪ {C}
```

$|S \cap S_i| \geq \lambda$ and $S_i \not\subseteq S$. Only these clusters are potentially collinear to C. If no such C_i exists, we can simply add C to the current set of $\lambda Clusters$. Otherwise, for each potentially collinear cluster C_i we do the following steps:

We generate the candidate $C_{merge} = (O \cap O_i, S \cup S_i, \lambda)$ and check if it is a valid cluster. If so, we first add C_{merge} to Tmp since it potentially can be merged with further clusters later on (Note: Tmp acts as a queue, where we successively remove and add elements; the method stops if Tmp is empty). Second, we check whether C_i is redundant to C_{merge}. Since the 'potentially redundant' rule is automatically fulfilled, we simply have to test $|O_i| - |O \cap O_i| < minSize$. If C_i is redundant, we remove it from $\lambda Clusters$, ensuring a manageable number of clusters at any time. Similarly, we check the redundancy of C w.r.t. C_{merge}. If it is redundant, we mark it correspondingly.

If each C_i is processed, i.e., its merging with C has been analyzed, we remove C from Tmp. If C is *not* marked as redundant, we finally add C to the set $\lambda Clusters$.

Note that the termination of our merging principle is guaranteed due to the condition $S_i \not\subseteq S$. Since the subspace cardinality of the merged clusters increase, at some point in time no further merging partners can be found, and the set Tmp

will become empty. Note also that our merging is invoked *within* the for-loop (line 4, Algorithm 5.1). Thus, the set Tmp is usually small and the clusters in $\lambda Clusters$ may already be of much higher subspace cardinality than $\lambda + 1$. Overall, our merging principle efficiently generates λ-dimensional clusters of subspace cardinality larger than $\lambda + 1$.

5.4.3 Object pruning exploiting the redundancy model

By our pruning step, we try to lower the number of objects of the current subspace S that have to be analyzed for their clustering structure. According to Definition 5.6, a non-redundant cluster C_S of dim. λ must have sufficiently high support even if the object sets of some other clusters (based on Def. 5.5) are removed. Since we already know the non-redundant clusters with dimensionality $\lambda' < \lambda$, we can select those clusters $C' \in Result$ that fulfill the 'potentially redundant' rule w.r.t. the current subspace S, i.e., those clusters C' for which $C_S \prec_{red}^{pot} C'$ holds[3]. Clusters fulfilling the rule might be the reason for *induced* correlation clusters in S and hence their object sets can safely be removed. To put it clearly: If C_S is a valid non-redundant cluster in S, it will still be so even after removing the objects determined above.

But we can remove even more objects: Our merging principle generates clusters with subspaces of cardinality larger than $\lambda+1$. For example, based on clusters in $\{d1, d2\}$ and $\{d1, d3\}$ we may get a cluster C_m in $\{d1, d2, d3\}$; it has the same cluster dimensionality λ but higher subspace cardinality. Thus, before analyzing the subspace $\{d2, d3\}$ we can also remove the objects of C_m since the 'potentially redundant' rule holds for this cluster. In general, we can prune objects contained in clusters $C' \in \lambda Clusters$ with $C_S \prec_{red}^{pot} C'$. This prevents the detection of redundant *collinear* clusters with lower subspace cardinality. Hence, our merging principle introduced in the previous paragraph also leads to an efficiency gain within this subroutine.

Formally, the set of non-pruned objects is given as

$$NonPruned_S = \{o \in DB \mid \neg\exists C' \in Result \cup \lambda Clusters : C' = (O', ., .) \wedge o \in O' \wedge (DB, S, |S| - 1) \prec_{red}^{pot} C'\}$$

[3]At this point we do not know which objects O are clustered in C_S. However, since Definition 5.5 just uses λ and S, and we have $\lambda = |S| - 1$, all necessary information is given. We can e.g., simply set $C_S = (DB, S, |S| - 1)$.

Please note that depending on the current subspace S different sets of objects are pruned since the 'potentially redundant' rule may be evaluated differently. Thus, objects can still contribute to several correlations due to different subspace projections as desired by our model.

Overall, for each subspace S a large amount of objects might be already removed based on the clusters in $Result$ (to prevent induced clusters) and $\lambda Clusters$ (to prevent collinear clusters). Only in the remaining set of objects novel clusters have to be detected.

5.4.4 Summary

Our SSCC generates clusters bottom-up w.r.t. their dimensionality and simultaneously removes redundant clusters top-down w.r.t. their subspace cardinality. Due to the redundancy model and based on the increasing set of known clusters, we are able to prune for each subspace a large amount of objects that have not be considered at all. Furthermore, by utilizing the collinearity phenomenon and merging several clusters, we can avoid analyzing many subspace projections. Overall, SSCC efficiently determines a non-redundant subspace correlation clustering result.

5.5 Experimental Analysis

Setup We compare SSCC against all (non-hierarchical) algorithms implemented in the framework ELKI [AKZ08], namely ORCLUS [AY00], COPAC [ABK+07b], and 4C [BKKZ04]. For subspace correlation analysis we choose CARE [ZPW08a]. Since REDUS [ZPW08b] is not exclusively designed for linear correlations, CARE is better suited for our settings. To revisit our statement of the introduction that it is not sufficient to simply use the groupings determined by correlation clustering and to calculate the truly correlated dimensions in a post-processing step, we implement a post-processing method that discards all dimensions being approximately parallel to the principal components of a cluster and recalculates the cluster's dimensionality. We name this step PP and apply it to all results of full-space correlation clustering methods.

We measure clustering quality by the CE measure (clustering error) [PM06], which also considers the subspaces in its evaluation. For easier interpretation we depict the results of 1-CE, where 1 indicates perfect quality and 0 lowest possible

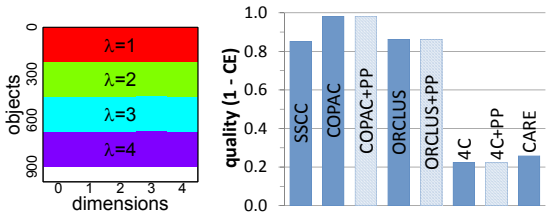


Figure 5.5: Scenario A

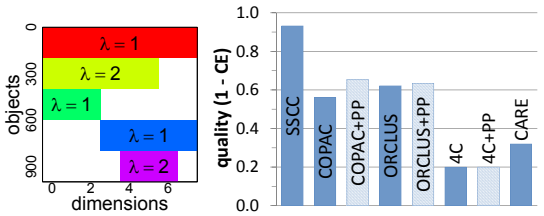


Figure 5.6: Scenario B

quality. Efficiency is measured by the approaches' runtime. For comparability all experiments were conducted on Opteron 2.3GHz CPUs using Java6 64bit.

We start by evaluating the approaches based on synthetic data to analyze their performance for the different subspace scenarios. We continue by observing the approaches' behavior for different database scalings and will confirm important observations for the real world datasets 'Wages'[4] and 'Image Segmentation' [FA10]. For each algorithm we determine optimal parameter settings w.r.t. the CE value.

Test scenarios for subspace clustering In Figures 5.5-5.10 we depict the algorithms' results for different correlation scenarios, especially for subspace settings. To allow a better interpretation, we provide visual descriptions of the used datasets attached to the left of the evaluation results of each particular test scenario. Each colored region corresponds to one cluster covering a specific set of objects in a specific subset of dimensions. λ indicates the cluster's dimensionality. Attribute values in white regions are noise.

For the first simple test scenario A (Figure 5.5), with only full-space clusters, the full-space approaches COPAC and ORCLUS perform better than SSCC. Since clusters are well separated in the full space but are likely to be merged in subspace projections, the merging strategy of SSCC tends to assign few objects to wrong clusters. Surprisingly, 4C does not yield good results, which probably is originated by its sensitivity to the setting of the neighborhood range, which however might be different for clusters of different dimensionality λ, and its requirement of spatially connectedness of the clusters. Since the CARE approach requires a cluster to comprise the majority of a dataset's objects, it does not perform well for datasets with more than one cluster per subspace, which will be confirmed by the other test cases.

[4] http://lib.stat.cmu.edu/datasets/CPS_85_Wages

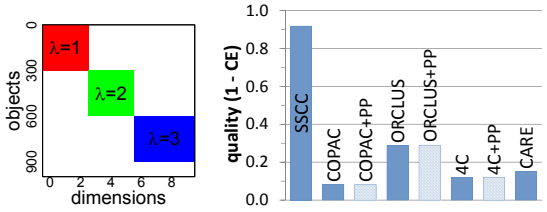


Figure 5.7: Scenario C

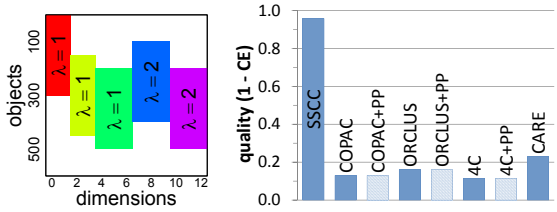


Figure 5.8: Scenario D

Test case B (Figure 5.6) shows a simple subspace scenario, where clusters are still disjoint and only few dimensions per cluster are noisy. Our SSCC manages to achieve even better clustering results for this setting. The results of COPAC and ORCLUS are significantly lower compared to scenario A. Noise dimensions obfuscate the clustering structure in the full-space and *even post-processing* only marginally improves the quality of the detected clusters. In contrast, CARE is able to achieve better results than for test case A, as clusters have less overlap per dimension.

Although scenario C (Figure 5.7) seems to be the easiest subspace setting, as neither dimensions nor objects of clusters do overlap, it poses severe challenges for full-space algorithms. Since the degree of noise dimensions exceeds the one of relevant dimensions for all clusters, COPAC, ORCLUS, and 4C do not manage to reveal the true clustering structure. Even post-processing achieves no further improvement. As discussed in the introduction, existing correlation clustering methods do not just suffer from their restriction to disjoint clusters (in Figure 5.7 the clusters do not overlap!) but they analyze the full-space where no clearly outstanding grouping is present. By analyzing (individual) subspace projections, SSCC increases the discrimination power between different correlations. SSCC is the only approach being able to uncover the correlations.

Contrarily to scenario B, for the clusters of scenario D (Figure 5.8) not the object sets but the subspaces are disjoint. As all approaches besides SSCC partition the data into clusters, they will not recover the clustering structure by design. However, since the clusters' overlap is arbitrary, the correlations interfere with each other and no clear statement about correlations is possible in the full space. Still, SSCC nearly perfectly recovers the hidden correlations.

Although the degree of overlap of clusters w.r.t objects is increased in scenario E (Figure 5.9) in comparison to scenario D, clustering results of COPAC and ORCLUS are better. Clusters either agree perfectly or not at all regarding their objects or

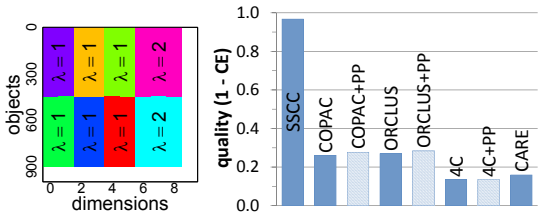


Figure 5.9: Scenario E

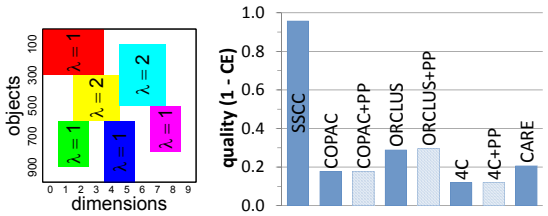


Figure 5.10: Scenario F

dimensions. Therefore, the projection into full-space still enables the detection of a certain clustering structure. Thus, not only the partitioning of the data hinders the cluster detection for full-space algorithms but also the arbitrary interference shown in scenario D.

The last test case F (Figure 5.10) shows a scenario typically described for subspace clustering. For these combined characteristics of the previous cases, we once more observe that none of the examined approaches but SSCC is able to reveal the underlying correlation clustering structure of the data.

For further experiments we will restrict our comparison of SSCC to COPAC and ORCLUS, as the remaining methods have shown to be not effective to typical subspace scenarios.

Quality w.r.t. number of noise dimensions To confirm our observations that irrelevant dimensions obscure the clustering structure in the full-space, in Figure 5.11 we gradually add noise dimensions to the dataset of Figure 5.5. While SSCC is mostly unaffected by noise dimensions, COPAC and ORCLUS increasingly fail to discover the hidden correlations. For few noise dimensions, post-processing

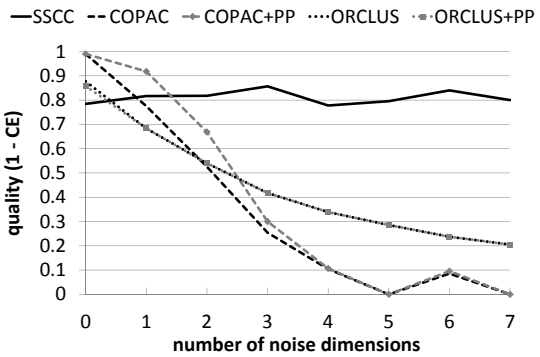


Figure 5.11: Effect of noise dimensions

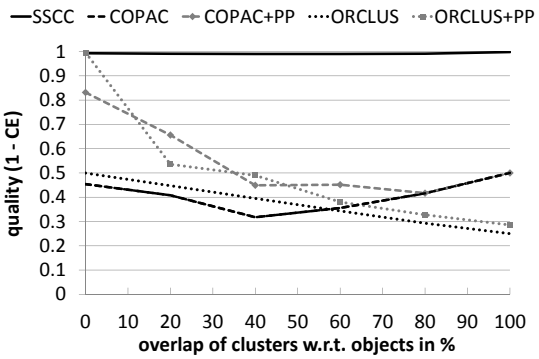


Figure 5.12: Overlapping clusters

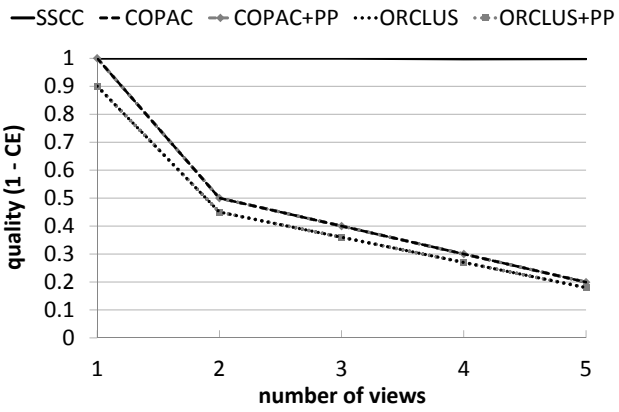


Figure 5.13: Effect of multiple views

slightly increases the quality of COPAC. In general, however, post-processing cannot eliminate the problems of full-space clustering.

Quality w.r.t. overlap of clusters In Figure 5.12 we examine the effect of increasing percentage of overlapping objects between clusters. For a 4d dataset with 1000 objects, 2 clusters each with 500 objects in disjoint 2d subspaces, we increase the clusters' overlap without varying the clusters' sizes. SSCC shows perfect result for any degree of overlap. For ORCLUS and COPAC the post-processing is able to reveal the true correlations for small overlap. However, with increasing overlap, both algorithms are destructed by the interference of the two clusters in the full-space, shown by worse post-processing quality. Even worse, COPAC groups the overlapping objects into a separate cluster, which is according to CE more beneficial than to regard these objects as noise.

Quality w.r.t. number of views In Figure 5.13 we systematically increase the number of views for a dataset of 1000 objects and 10 dimensions. Comparably to scenario E (5.9), all views have disjoint, nearly equally sized subspaces and contain two one-dimensional correlation clusters. We observe a fast decrease of clustering quality of COPAC and ORCLUS with increasing number of views. SSCC remains unaffected by the view number, showing perfect results.

Scalability w.r.t. database size Figure 5.14 (left) shows the results for varying number of objects for a dataset with 5 dimensions and 4 equally sized full space

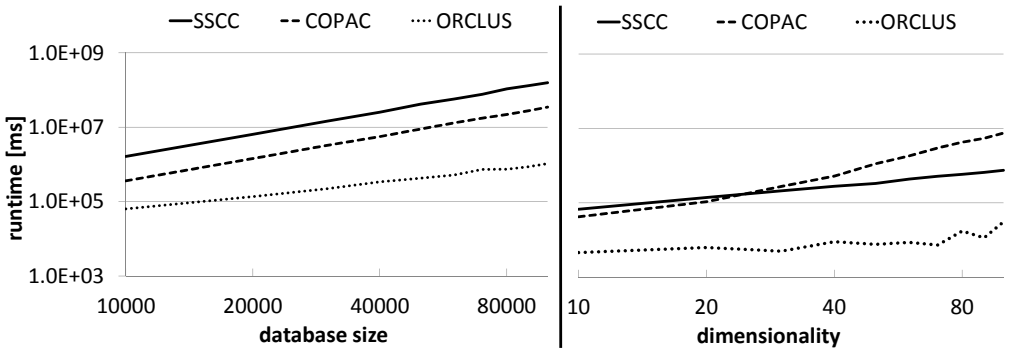


Figure 5.14: Scalability w.r.t. database size & dimensionality

clusters without noise. All three algorithms scale linearly with the database size. Note the logarithmic scale of *both* axes. Although SSCC has to cope with an exponential number of subspaces, its runtime is still in range of COPAC's runtime.

Scalability w.r.t. dimensionality For a dataset similar to scenario E (Figure 5.9), except that all clusters comprise 500 objects, we consecutively concatenate the dataset to generate datasets of higher dimensionality in Figure 5.14 (right). Although the number of subspaces grows exponentially with the number of dimensions, SSCC scales linearly. COPAC shows super linear behavior and thus exceeds SSCC's runtime.

Real world data In the following we analyze the results of SSCC and COPAC for real world data. For *Wages* (534 objects, 4 numerical attributes, 7 categories) we only used the four numerical attributes: 'education', 'experience', 'wage', and 'age'. Both algorithms detected only a single 2-dimensional cluster in the subspace of 'education', 'experience', and 'age' as visualized in Figure 5.15. Note that COPAC does not actually report the relevant subspace but we have to use the post-processing step to exclude the uncorrelated dimension 'wage'. In Figure 5.15, all clustered objects are colored red, noise is colored blue. For the 2d correlation in the Wages data, SSCC captures the objects much better than COPAC. COPAC misses many objects since in the full-space they do not belong to this correlation, i.e. the additional attribute 'wage' disturbs the local correlation of these objects. By analyzing subspace projections, SSCC is able to detect the truly correlated set of objects.

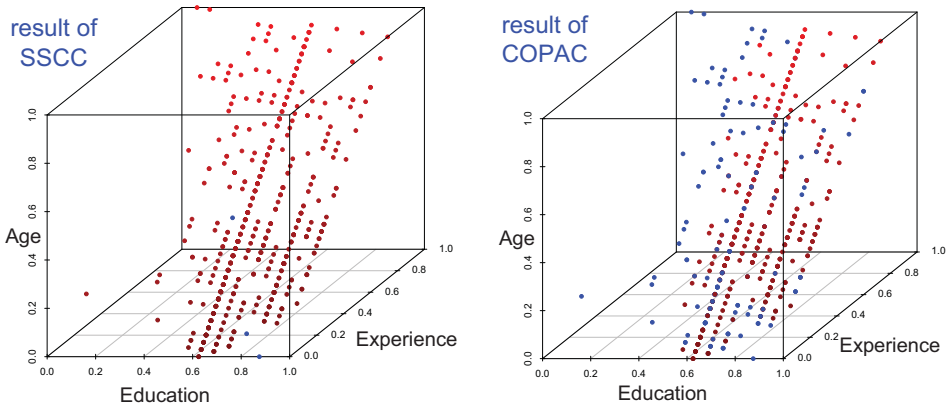


Figure 5.15: Results of SSCC and COPAC for Wages

For the *Image Segmentation* data we have 19 numerical attributes describing pixel regions and one class attribute. We remove the class attribute and the constant region-pixel-count attribute and normalize the remaining attributes between zero and one. Although trying a wide range of parameter settings, COPAC was not able to detect any cluster for this dataset. SSCC detected eight clusters, and two exemplary clusters are plotted in Figure 5.16. Obviously, the clusters detected by SSCC correspond to strong correlations, which can be visually verified and are explainable from the dataset's description. For example, SSCC is able to detect a cluster in the subspace of $exgreen.mean$, $exblue.mean$, and $value.mean$ (cf. Figure 5.16 (right)). According to the dataset's description, the first two dimensions are the measures of excess green and excess blue that are defined as $exgreen.mean = (2G + (R + B))$ and $exblue.mean = (2B + (R + G))$. Here R de-

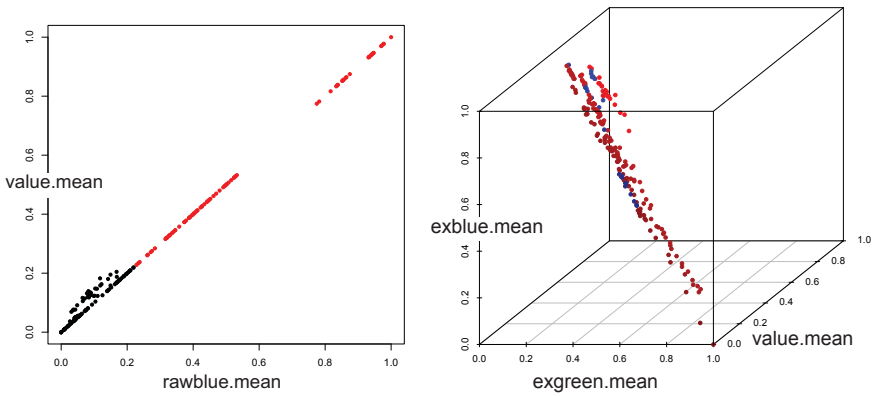


Figure 5.16: SSCC results for Image Segmentation

notes the average over the raw values of red, B for blue, and G for green, respectively. The dimension $value.mean$ is a 3-dimensional non-linear transformation of the values R, G, and B. Obviously, from the definition of these dimensions we can expect some correlation in this subspace. This correlation is successfully identified by SSCC. Note that all of the correlations detected by SSCC appear only in subsets of the attributes. Thus, full-space approaches are not able to detect these results, confirming the need of our novel subspace correlation clustering paradigm.

5.6 Conclusion

In this chapter we have demonstrated that the observations of traditional subspace clustering analogously apply for the paradigm of correlation clustering. Due to the decreasing discrimination power of distances, which results from overlapping clusters and irrelevant dimensions, a clustering conducted in the full-space is questionable. Thus, even a post-processing to refine the clustering result and to exclude uncorrelated dimensions is not sufficient for typical subspace scenarios. Instead, we analyze *subspace projections* of the data to find meaningful *strong correlations* supported by *subsets of objects*. Our introduced approach reveals linear correlations in subspace projections, allows objects to contribute to multiple correlations, and simultaneously ensures a result of manageable size containing only non-redundant subspace correlation clusters. For this, we carefully differentiate between non-redundant correlation clusters and ones originated from collinearity or induction. We design the efficient algorithm SSCC exploiting various pruning techniques. As demonstrated in the experiments, a transfer of the ideas and observations from the subspace clustering paradigm leads to more precise correlation clustering results compared to state of the art techniques in this domain.

With this chapter we conclude the first part of this thesis. We introduced novel subspace clustering techniques that solve the challenge of redundancy, are able to detect multiple views, and extract dense areas or correlations from the data. In the two subsequent parts of this thesis we use the gained insights to develop subspace clustering methods going beyond vector data but handling more complex data types.

Part II

Subspace Mining on Imperfect Data

Chapter 6

Imperfect Data: Introduction

While the previous part of this thesis has introduced subspace clustering methods handling preprocessed data, the following part introduces subspace clustering techniques tackling the challenge of imperfect information. Before presenting two solutions for different kinds of imperfect information in Chapter 7 and 8, we start with a classification of the different types of imperfect information and we discuss sources for their occurrence in real world scenarios. To highlight the need for mining methods that directly handle imperfect data, we give a brief overview of data cleansing principles and we show their general drawbacks.

6.1 Types of Imperfect Information

Several schemes to classify imperfect information were proposed in the literature. So far, however, no standardized classification scheme is available. While [BT85] uses the terms *incompleteness*, *imprecision*, and *uncertainty*, the methods of [BP96, Tzv90] refer to *imprecision*, *uncertainty*, *vagueness*, and *inconsistency*. In [Par96] a thorough discussion of the different terms is presented and current methods to handle such data are reviewed. In the following, we focus on the most important types of imperfect information.

Incomplete data. Incompleteness describes imperfect information due to the absence of individual measurements. While the absence of specific information regarding a single object is denoted as existential incompleteness, the absence of objects as a whole is denoted as universal. Incomplete information, often referred

to as data containing *missing values*, occurs for example due to faulty sensors or incomplete customer questionnaires.

Imprecise data. In contrast, imprecision describes imperfect information that is recorded but without sufficiently high granularity. For example, the age of a customer is just given by an interval or by a fuzzy description as ”young”. The notion of imprecision depends on the method that is used to analyze the recorded data: for some tasks the granularity of the measured data is sufficient, e.g. just the year of birth is needed, while other tasks require more precise data, e.g. the exact day of birth. In general, imprecision is present if the attributes of an object are non-singleton subsets of the domain to be analyzed. Intuitively, if an attribute covers the whole domain, it corresponds to a missing value, e.g. the age of a customer is in the range 0 to 200.

Inconsistent data. Inconsistency, as a third type of imperfect information, describes the occurrence of conflicting values within a database. For example the city and the state of a customer do not match. This type of error often occurs if different sources of information are used and merged into single database.

Uncertain data. Finally, in many scenarios uncertainty about the given information does exist. In the case of uncertainty, one is just provided with an estimate how likely the observed value is equal to (or may differ from) the true value. For example, the measured GPS signal of a mobile phone is highly uncertain information for determining its true position and one is only provided with an estimate about this position by, e.g., incorporating probability distributions. Similar to incomplete information, one distinguishes uncertainty about specific attributes – so called attribute uncertainty – and uncertainty about the existence of whole objects – tuple uncertainty. Besides uncertainty due to the data recording step, artificial uncertainty due to privacy issues is present, i.e. before providing a data set sensitive information is obfuscated.

As mentioned in [Par96], uncertainty differs to the previous types of imperfect information since it is inherently subjective. An external subject, e.g. the user, has to specify the underlying model, e.g. the probability distribution, how to estimate the truth of a fact. The previous types, however, are objective since they are elementary results of the data’s recording process. Furthermore, uncertainty and imprecision are often closely connected since, e.g., a ”young” customer is more likely 25 than 50 years old.

6.2 Data Cleansing

As shown in the previous section, different types of imperfect information exists and usually they cannot be handled by traditional data mining methods. Thus, data cleansing techniques try to correct or remove imperfect information [HK01]. Various data cleansing methods were introduced ranging from simple parsing to detect incorrect strings, over data transformations to convert measurements to different scales, to duplicate elimination to identify (almost) identical entries. More advanced techniques use statistical methods, e.g, to impute missing values or they perform outlier analysis to detect erroneous measurements.

Even though all these techniques try to increase the data quality, they have several limitations. First, data cleansing is accompanied by high cost since the methods are rarely completely automatic but the user has to be involved. Second, the storage overhead can be huge since besides the original data also the preprocessed data have to be stored. And last, preprocessing the data usually results in an information loss. On the one hand the preprocessing step is not aware of the special characteristics of the subsequent data mining method, and on the other hand the mining method cannot distinguish between a precise object and an imperfect but cleaned object.

Overall, valuable information is no longer available due to preprocessing. Especially for the task of subspace clustering, as we will show in the next chapters, this can lead to undesired clustering results. Thus, our aim is to develop integrated mining methods that direct handle imperfect data for the task of subspace clustering as illustrated in Figure 1.4. While our first method provides a solution for the case of data containing incomplete information, our second method covers the topic of attribute uncertainty.

Chapter 7

Subspace Clustering for Incomplete Data

In today's applications, data analysis tasks are hindered by many attributes per object as well as by imperfect data containing missing values. Subspace clustering tackles the challenge of many attributes by cluster detection in any subspace projection of the data. However, it poses novel challenges for handling missing values of objects, since each object can be grouped in multiple clusters due to locally different subspace projections (cf. Figure 7.1).

	d1	d2	d3	d4	d5
o1	low value	low value	low value		
o2	low value	low value	low value		
o3	low value	low value	low value		
o4	low value	low value	???	high value	high value
o5			high value	high value	high value
o6			high value	high value	high value
o7			high value	high value	high value

Figure 7.1: Simplified example of a missing value in multiple subspace clusters

Intuitively, missing values should be tolerated in such cases where sufficient information is given about the relevant dimensions and the object groupings. Thus, in Figure 7.1 for example the object $o4$ should be included in both subspace clusters. This general idea is independent of the underlying subspace cluster definition. Unfortunately, it is not supported by traditional techniques that handle missing values.

In this chapter, we propose a general fault tolerance definition enhancing sub-

space clustering models to handle missing values. We introduce a flexible notion of fault tolerance that adapts to the individual characteristics of subspace clusters and ensures a robust parameterization. Allowing missing values in our model increases the computational complexity of subspace clustering. Thus, we prove novel monotonicity properties for an efficient computation of fault tolerant subspace clusters. Experiments on real and synthetic data show that our fault tolerance model yields high quality results even in the presence of many missing values.

7.1 Motivation and Comparison with Related Work

Limitations of statistical pre-processing methods. The easiest way to handle missing values is by eliminating objects containing missing values, the so called complete case analysis [LR87]. It is known that this principle leads to huge information loss and may distort the distribution of the data. As an alternative, imputation techniques were proposed to fill up the missing values. Single imputation techniques such as imputing by means, hot deck imputation, or imputation based on maximizing the observed-data likelihood [SG02a], fill up missing values with a single value. By using just a single value, we would miss one of the hidden clusters in Fig. 7.1. Subspace clustering, however, would be able to cope with multiple different values depending on the used subspace projection.

Multiple imputation [Sch97, ACHM02] was introduced by [Rub87] and, for example, applies Monte Carlo techniques to determine the values used for imputation. The general idea of multiple imputation is to generate several (complete) data sets. The missing values are, e.g., based on simulation or drawn from a learned distribution. The used distribution is identical for any of the generated data sets, but in our scenario missing values should be handled based on the current context, i.e. which subspace is currently considered (cf. Fig. 7.1). A more severe problem of multiple imputation is that we explicitly have to handle several data sets: for each data set subspace clustering has to be performed and the results have to be joined (pooling step). Thus, we need a technique for combining different subspace clustering results to obtain an overall solution. While for traditional clustering ensemble/consensus techniques are available for this task [SG02b, BS04], for subspace clustering there exists no such consensus function to combine multiple clustering results. Furthermore, naive computation of independent subspace clustering results on these complete data sets would be highly

inefficient. Thus, current subspace clustering techniques cannot handle multiple data sets as generated by multiple imputation.

Overall, all statistical imputation techniques act as pre-processing methods that are used before the actual clustering step. Our approach, in contrast, handles missing values during the clustering process, i.e. we develop an integrated mining method. Even though the statistical pre-processing techniques have drawbacks, we include complete case analysis and single imputation, both in combination with existing subspace clustering techniques, as competitors in our experimental analysis.

Missing value handling in related mining paradigms. While there exist fault tolerant extensions for the (subspace clustering related) mining paradigms of itemset mining [YFB01, PG09] and for the bi-clustering/co-clustering related formal concept analysis [PB05, BRB06], they are not directly applicable for subspace clustering. All of these approaches were developed for binary data while for subspace clustering we analyze numerical data. Also sets of similar objects are not adequately considered in previous works. For example in fault tolerant itemset mining only the support of an itemset is determined, not its actual object set as in subspace clustering. The similarity of the object sets, however, is the reason for redundancy and hence for fault tolerant subspace clustering we have to cope with this challenge. Only one subspace clustering publication mentions the challenge of missing values [AGGR05], but it does not provide a fault tolerant solution. Simply allowing missing values in the relevant dimensions would introduce pseudo clusters not present in the data, while simply discarding missing values as irrelevant dimensions would lose some meaningful clusters.

Our contribution. As contribution of our fault tolerant subspace clustering model we tackle major open challenges in fault tolerance. Overall, we develop:

- A general fault tolerance model handling missing values in subspace clusters
- A flexible and robust notion of fault tolerance adapting to cluster characteristics
- Novel monotonicity properties enabling efficient candidate pruning

We abstract from concrete subspace clustering definitions and propose a general fault tolerance principle applicable to multiple instantiations. Thus, grid-based subspace clustering approaches as CLIQUE [AGGR05] and its enhancements [NGC01, SZ04, PJAM02, YM03], paradigms based on the density-based clustering idea [KKK04, KKRW05, AKMS08a], and several other definitions can benefit from

our approach. In addition to our general model, we present a concrete instantiation to the well-established grid-based subspace clustering.

Fault tolerance introduces three novel challenges in both modeling of subspace clusters and their computation: First, we have to take care that each object can be part of multiple subspace clusters due to different subspace projections. Second, a meaningful fault tolerance has to consider the varying object and attribute characteristics for each subspace cluster individually. Therefore, we introduce a fault tolerance notion that adapts to the characteristics of subspace clusters. To quantify the amount of acceptable missing values we introduce parameters and we compare different models of thresholds. Third, the exponential search space of all possible subspaces poses a general challenge for subspace clustering. Allowing missing values introduces an additional search space complexity. Thus, pruning techniques for our fault tolerant model are essential to enable an efficient computation. We prove monotonicity properties for our thresholds and show especially the robustness of our flexible fault tolerance in the empirical evaluation. Overall, these challenges go beyond the ones of traditional subspace clustering. We tackle these challenges by our general fault tolerance model, which achieves high clustering quality even in the presence of many missing values.

7.2 Fault Tolerant Subspace Clustering Model

In this section, we present the theoretical basics for tackling the challenges of missing value handling in subspace clustering. The basic principle of the model is to permit a cluster to contain a specific amount of missing values. This way, we want to ensure that all hidden clusters are found, although some of the inner structure of the dataset is obfuscated by missing values. We first present our general model and propose a specific instantiation in Section 7.3.

7.2.1 Relaxation of subspace cluster definitions

In general, a subspace cluster $C = (O, S)$ is a set of objects $O \subseteq DB$ together with a set of dimensions $S \subseteq Dim = \{1, \ldots, n\}$. In the given database, each object $o \in DB$ corresponds to an n-dimensional feature vector $o = (o_1, \ldots, o_n)$ such that $DB \subseteq (\mathbb{R} \cup \{?\})^n$ with "?" as a missing value. Instead of requiring similarity in the n-dimensional space, a subspace cluster (O, S) assumes similarity of

objects O only w.r.t. a set of dimensions S. Thus, a subspace cluster definition poses similarity constraints only based on these dimensions. Depending on the underlying cluster model one, e.g., uses the Euclidean Distance in subspace S, i.e. $dist_S(o,p) = \sqrt{\sum_{d \in S}(o_d - p_d)^2}$, or poses constraints at each dimension individually by restricting $|o_d - p_d|$ to a maximal deviation for all $d \in S$. All other dimensions $d' \notin S$ are irrelevant for this cluster and do not pose any constraints to the object set O.

Definition 7.1 (Constrained Attribute Values)
For a subspace cluster $C = (O, S)$ a clustering model poses constraints for each object $o \in O$ in all relevant dimensions $d \in S$: o_d is called a constrained attribute value of (O, S) iff $(o, d) \in O \times S$

A missing value would affect a subspace cluster (O, S) if it occurs in one of the constrained attribute values. Traditional subspace cluster definitions require all of these constraints to be fulfilled. Thus, they are not able to handle missing values. Our fault tolerance model allows to relax some of these constraints to allow missing values in the relevant dimensions S of clustered objects O. With our fault tolerance model, any traditional subspace cluster definition can be relaxed to allow missing values. For example, one can use grid-based [AGGR05, NGC01, SZ04, PJAM02, YM03] or density-based [KKK04, KKRW05, AKMS08a] cluster definitions. For each missing value one simply assumes to have given the ideal value instantiation inside the subspace cluster. Thus, only the non-missing values have to fulfill the cluster definition while missing values relax the cluster definition. Unfortunately, this naive relaxation to fault tolerance would allow any amount of missing values. Trivial clusters consisting only of missing values would be detected as valid subspace clusters. Our fault tolerance model enhances this loose relaxation. The key idea is to restrict the number of allowed missing values such that the given data is sufficient to infer the missing parts. Hence, the non-missing values of a subspace cluster (O, S) are sufficient to validate the relevant dimensions S and clustered objects O.

Let us sketch this idea with a toy example: In Figure 7.2 a database with missing values is illustrated on the left. On the right, two potential subspace clusters are selected. Both subspace clusters cover the same relevant dimensions but include different objects. The upper cluster is not meaningful since the object 5 has just missing values in the relevant dimensions. Furthermore, the dimensions 2

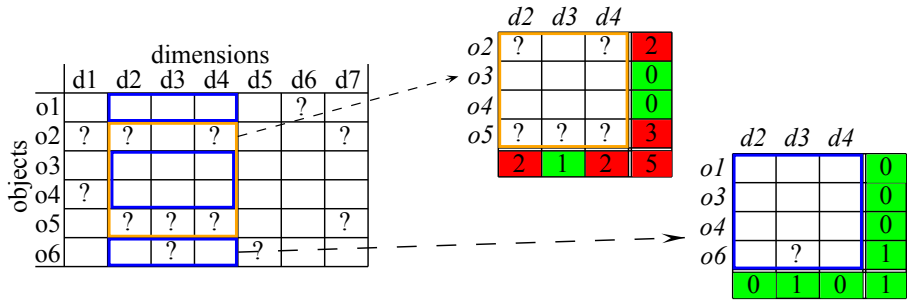


Figure 7.2: Database with missing values and two potential subspace clusters

and 4 have a high amount of missing values for this cluster; the relevance of these dimensions is not well justified. The lower cluster, however, contains just one missing value. Thus, it provides sufficient information. To check this formally, we first introduce a definition to retrieve the amount of missing values per dimension or per object of the cluster.

Definition 7.2 (Missing value set functions)
The objects having a missing value in dimension $d \in Dim$ *are defined by* $O_?(d) = \{o \in DB \mid o_d = ?\}$.
The dimensions of an object $o \in DB$ *containing a missing value are defined by* $D_?(o) = \{d \in Dim \mid o_d = ?\}$.

These functions return sets based on the full space and the whole database. However, we are only interested in those missing values lying in a potential cluster $C = (O, S)$, i.e. missing values corresponding to the constrained attribute values. Missing values in the irrelevant dimensions $Dim \backslash S$ or contained in the currently not considered objects $DB \backslash O$ do not have any impact on the decision whether (O, S) is a valid cluster or not. Thus, our model considers missing values based on the individual contexts, i.e. which subspace and object set is currently considered. Formally, we have to intersect the sets introduced in Definition 7.2 with the (potential) cluster: For the upper cluster in Figure 7.2 we get $O_?(3) \cap O = \{5, 6\} \cap \{2, 3, 4, 5\} = \{5\}$ and $D_?(2) \cap S = \{1, 2, 4, 7\} \cap \{2, 3, 4\} = \{2, 4\}$.

7.2.2 Novel fault tolerance bounds

The main idea is allowing a cluster to contain a certain amount of missing values, which has a big impact on the clustering result. Allowing no missing values would

end up in traditional subspace clustering that misses some clusters due to missing values. Allowing an arbitrary amount of missing values, however, includes pseudo clusters due to the loose model relaxation.

We propose three different characteristics to restrict the allowed number of missing values in a meaningful way:

(1) Each object of the cluster should not contain too many missing values since otherwise the similarity to the remaining objects is questionable. We denote this property **object tolerance** since it is checked for each object of the cluster.

(2) Each dimension should not contain too many missing value since otherwise it is questionable whether this dimension is relevant for this cluster. This property is called **dimension tolerance**.

(3) We restrict the overall number of missing values in our cluster, i.e. by considering the objects and dimensions simultaneously: the **pattern tolerance**.

As first tolerance properties we introduce constant bounds as follows. More enhanced variable bounds are defined in the remainder of this section.

Definition 7.3 (Fault tolerances (constant case))
Given the tolerance thresholds $\varepsilon_o^c, \varepsilon_s^c, \varepsilon_g^c \in \mathbb{N}_0$, a subspace cluster $C = (O, S)$ is

- *ε_o^c–object tolerant, if* $\forall o \in O: \ |S \cap D_?(o)| \leq \varepsilon_o^c$
- *ε_s^c–dimension tolerant, if* $\forall d \in S: \ |O \cap O_?(d)| \leq \varepsilon_s^c$
- *ε_g^c–pattern tolerant, if* $\sum_{o \in O} |S \cap D_?(o)| \leq \varepsilon_g^c$

By selecting $\varepsilon_o^c{=}2, \varepsilon_s^c{=}1, \varepsilon_g^c{=}4$ in Figure 7.2 the upper cluster violates all thresholds while the lower cluster fulfills all of them. If we chose $\varepsilon_g^c = 6$, the pattern tolerance would be fulfilled by the upper one; however, the remaining two tolerances are still violated. Note that none of these properties subsumes one of the others. For example, one might be willing to accept at most 2 missing values per object but at most 10 for the *whole* cluster. If the cluster covers more than 5 objects, some objects must have less than 2 missing values. In this case, the ε_o^c threshold cannot be reached by all objects simultaneously since by the ε_g^c–pattern tolerance we take a global view of all objects. Thus, all three thresholds are beneficial.

If a cluster fulfills all three tolerances, a sufficient amount of information can be used to validate the underlying subspace cluster definition. Thus, assuming

an ideal setting for the missing values to obtain a valid grouping is reasonable. Besides this advantage of Definition 7.3, the drawback is the constant and thus fixed number of permitted missing values. Though, the subspace clusters hidden in the data can differ w.r.t. their number of objects as well as their number of relevant dimension. Working with constant thresholds, however, forces each cluster to keep conditions regardless of its dimensionality and size. Hence, a 4-dimensional cluster is allowed to have the same amount of missing values as a 16-dimensional one. For example, setting $\varepsilon_o^c = 1$ allows each object to cover one dimension with missing values. This is meaningful for the 4-dimensional cluster but too restrictive for the 16-dimensional one.

Formally, this challenge of flexible fault tolerance can be derived out of the number of constrained attribute values (cf. Def. 7.1). As motivated in the previous paragraph, each dimension poses additional constraints to the objects in a subspace cluster (O, S). Thus, a constant fault tolerance (cf. Def. 7.3) would fail with increasing number of dimensions. Similarly, for increasing cluster size the same drawback of constant tolerance can be observed. Overall, more and more constraints have to be fulfilled due to the underlying subspace cluster definition. Hence, the fault tolerance has to adapt w.r.t. the characteristic properties size $|O|$, dimensionality $|S|$ and number of constrained attribute values $|O \times S|$.

We propose using relative thresholds to solve this problem. The key idea is to limit the ratio of missing values and attribute value constraints. Thus, with increasing number of constraints one allows a proportional increase of missing values to be tolerated. Our flexible fault tolerance adapts to the dimensionality and size of a pattern: the characteristics of the cluster are taken into account. For example, a *relative object tolerance* of $1/4$ allows each object of the current subspace cluster $C = (O, S)$ to cover $1/4 \cdot |S|$ dimensions with missing values. Thus, for the 4-dimensional cluster at most one missing value per object is possible and for the 16-dimensional cluster at most 4 missing values per object. Thus, relative thresholds are more suitable than constant thresholds.

Definition 7.4 (Fault tolerances (relative case))
Given the tolerance thresholds $\varepsilon_o, \varepsilon_s, \varepsilon_g \in [0, 1]$, a subspace cluster $C = (O, S)$ is

- *ε_o–object tolerant, if* $\forall o \in O : \ |S \cap D_?(o)| \leq \varepsilon_o \cdot |S|$
- *ε_s–dimension tolerant, if* $\forall d \in S : \ |O \cap O_?(d)| \leq \varepsilon_s \cdot |O|$
- *ε_g–pattern tolerant, if* $\sum_{o \in O} |S \cap D_?(o)| \leq \varepsilon_g \cdot |O| \cdot |S|$

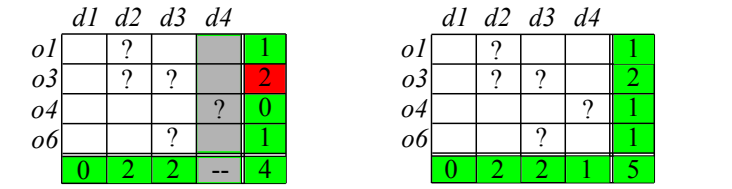


Figure 7.3: Relative thresholds: invalid cluster (left) and valid cluster (right) with $\varepsilon_o = \varepsilon_s = 0.6$ $\varepsilon_g = 0.5$

In Figure 7.3 two clusters covering the same objects but having different dimensions are illustrated. While the left one violates the 0.6-object tolerance property, the right one fulfills all thresholds since in the additionally included dimension 4 the object 3 does not contain a missing value. In contrast to the constant thresholds that ignore the increased proportion of given values, our relative threshold successfully adapts to the included dimension.

Overall, a valid fault tolerant subspace cluster is formalized by the following definition.

Definition 7.5 (Valid subspace cluster)
A cluster $C = (O, S)$ is called a valid fault tolerant subspace cluster, if

- *(O, S) fulfills all three tolerances (constant or relative)*
- *there exists an instantiation of the missing values from (O, S) such that the resulting cluster (O', S) (without missing values) fulfills the selected subspace cluster definition (e.g. [AGGR05, NGC01, KKK04, AKMS08a])*

Thus, depending on the currently selected cluster and subspace, a missing value can be instantiated differently enabling an object to be part of multiple clusters (cf. Figure 7.1). In the following we will focus on the flexible fault tolerance defined by the relative case and only briefly discuss the simpler constant case.

7.2.3 Proving monotonicity of fault tolerance

We prove some important monotonicity properties of our model. Most subspace clustering models obey a (anti-)monotonicity: if (O, S) is a subspace cluster, then in each subspace of S there exists a *superset of objects* such that this set is again a subspace cluster. Formally,

$$(O, S) \text{ cluster } \Rightarrow \forall S' \subseteq S : \exists O' \supseteq O : (O', S') \text{ cluster} \tag{7.1}$$

This monotonicity is an essential property for pruning the exponential search space in subspace clustering algorithms. The contrapositive of Equation (7.1) is used in a bottom-up processing to prune higher-dimensional subspaces. Unfortunately, as illustrated by the example in Figure 7.3, our fault tolerance model does not fulfill this monotonicity property. By considering variable thresholds we defined an enhanced model, however, if (O, S) is a valid subspace cluster this does not imply that (O', S') is one, too.

In the following, we derive a monotonicity property for enclosing cluster approximations of our fault tolerant model. These properties can later be used to increase the efficiency of algorithms (e.g. in our instantiation; Sec. 7.3.2). Let us first compare traditional monotonicity properties with our enclosing approximation: For monotone models, object sets increase in their size with decreasing dimensionality. This aspect is illustrated in Figure 7.4 (left) and the property holds for a variety of definitions including [AGGR05, NGC01, KKK04, AKMS08a]. Our fault tolerance model does not obey such a monotonicity. Thus, we propose an artificial monotonicity based on enclosing approximations, which correspond to supersets of the actual clusters (illustrated by rectangles in the figure). Although the clusters do not obey a monotonicity, our enclosing approximations do. In this case, Equation (7.1) is not directly valid for the clusters but valid for their enclosing approximations, which are later on refined to generate the actual clusters.

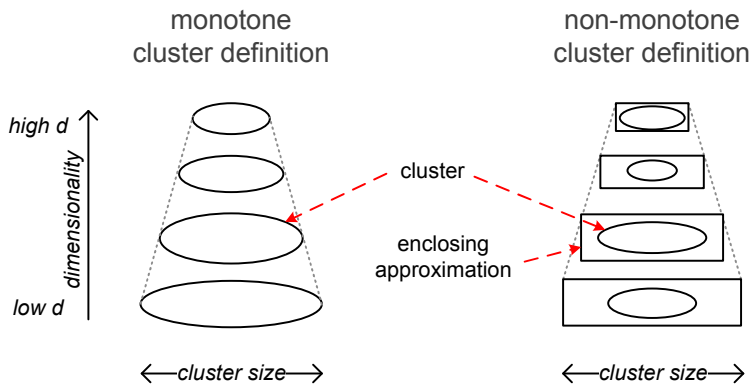


Figure 7.4: Monotonicity of clusters and enclosing cluster approximations

In the following, we prove several novel monotonicity properties for our fault tolerant subspace clustering model. To achieve this aim, we first have to develop enclosing approximations for fault tolerant subspace clusters, i.e. we have to define supersets of the (truly) clustered objects. Second, we show for these approximations certain monotonicity properties, which can be used for pruning the search

space. As a major aspect, we can prove that the monotonicity of Equation (7.1) holds for our approximations: If there appears no approximation in the subspace S', then we do not need to analyze any subspace $S \supset S'$. Overall, our fault tolerance model is applicable for several subspace cluster definitions. In our proofs we only assume the underlying subspace cluster model to fulfill Equation (7.1). Based on this, we prove that our extension to fault tolerance achieves a monotonicity by enclosing approximations as well.

Let us define our approximation for fault tolerant subspace clusters based on variable thresholds. The definitions for the simpler constant case can be derived respectively. The relative case is more complex since we have to consider the dynamic adaption of the thresholds to the characteristics of the clusters. In higher-dimensional spaces for example, we can permit a larger amount of missing values; thus, the corresponding clusters can potentially contain more objects compared to their lower dimensional counterparts. This aspect has to be considered by a monotone and enclosing cluster approximation.

We derive so-called mx-approximations. Intuitively, a subspace cluster in subspace S can only be extended by a few additional dimensions up to a dimensionality of mx (e.g. $mx = 4$ in Fig. 7.3). This maximal dimensionality is used to derive our enclosing approximation.

Definition 7.6 (Cluster approximation (relative case))
Let $\varepsilon_o, \varepsilon_s, \varepsilon_g$ be the selected tolerance thresholds. An mx-approximation $A = (O_A, S)$ is a maximal fault tolerant subspace cluster determined based on the adapted tolerance thresholds $\varepsilon'_o = \min\{\varepsilon_o \cdot \frac{mx}{|S|}, 1\}, \varepsilon'_s = \varepsilon'_g = 1$.

Intuitively, for an mx-approximation the dimension tolerance and the pattern tolerance are set to their maximal values. That is, an arbitrary number of missing values with respect to these two characteristics is possible. However, we still exploit the object tolerance based on the adapted threshold ε'_o. Thus, we actually do not include arbitrary objects in the approximation but objects with too many missing values must not be included. We show that each fault tolerant subspace cluster can be represented by a corresponding mx-approximation if the cluster's dimensionality does not exceed mx:

Theorem 7.1 *For each fault tolerant subspace cluster $C = (O, S)$ (w.r.t. $\varepsilon_o, \varepsilon_s, \varepsilon_g$) with dimensionality $|S| \leq mx$ exists an* enclosing *mx-approximation; that is, there exists an mx-approximation $A = (O_A, S)$ with $O_A \supseteq O$.*

Proof 7.1 *An mx-approximation is a fault tolerant subspace cluster based on relaxed thresholds. The dimension and pattern tolerance are set to the maximal values. The former object tolerance ε_o is also replaced by a larger value ε'_o since for all $|S| \leq mx$ it holds: $\min\{\varepsilon_o \cdot \frac{mx}{|S|}, 1\} \geq \min\{\varepsilon_o \cdot \frac{mx}{mx}, 1\} \geq \varepsilon_o$. Since $C = (O, S)$ fulfills all tolerances w.r.t. $\varepsilon_o, \varepsilon_s, \varepsilon_g$, it obviously fulfills all tolerances if the thresholds are relaxed. Since an approximation is a* maximal *fault tolerant subspace cluster (w.r.t. the relaxed thresholds), there exists an approximation $A = (O_A, S)$ with $O_A \supseteq O$.* □

The important aspect to be considered is the dependency between the dimensionality of the clusters and the mx-approximation. By selecting mx, the approximations are only correct for clusters up to this maximal dimensionality. Consequently, for a single cluster different approximations based on different mx values are possible. The smaller mx, the tighter is the approximation; however, with a small mx we account only for few subspace clusters since $|S| \leq mx$ is required. By fixing mx, the introduced cluster approximation shows also monotone behavior w.r.t. the object sets.

Lemma 1 *Monotonicity of objects*
Given an mx-approximation $A = (O_A, S)$ with $|S| \leq mx$, for each $S' \subseteq S$ there exists $O_B \supseteq O_A$ such that $B = (O_B, S')$ is again an mx-approximation.

Proof 7.2 *For each mx-approximation (O_A, S) in the subspace S we get the object tolerance condition $|S \cap D_?(o)| \leq \min\{\varepsilon_o \cdot \frac{mx}{|S|}, 1\} \cdot |S| \Leftrightarrow |S \cap D_?(o)| \leq \min\{\varepsilon_o \cdot mx, |S|\}$. Since always $|S \cap D_?(o)| \leq |S|$ this can be replaced by $|S \cap D_?(o)| \leq \varepsilon_o \cdot mx$. Thus, we get on the right hand side the constant term $\varepsilon_o \cdot mx$, and $O_A \subseteq O_B$ can be proven by contradiction: Assume there exists an object $o \in O_A$ with $o \notin O_B$. o cannot violate the object tolerance in O_B, since $\forall o \in O_A : \; |S \cap D_?(o)| \leq \varepsilon_o \cdot mx \overset{S' \subseteq S}{\Rightarrow} |S' \cap D_?(o)| \leq \varepsilon_o \cdot mx \Rightarrow$ object tolerance fulfilled. The dimension and pattern tolerance are always fulfilled for a cluster approximation. Consequently, only if o is not a valid element according to the cluster definition we can get $o \notin O_B$. However, we assume Equation 7.1 (anti-monotonicity) to be true for the cluster definition (this property was proven for, e.g., [AGGR05, NGC01, KKK04, AKMS08a]). Thus, there exists an O_B such that (O_B, S') fulfills the cluster definition. $\Rightarrow o \in O_B$ and hence $O_A \subseteq O_B$.* □

Besides the monotonicity of the object sets, we prove monotonicity properties of the cluster sizes of the potential clusters represented by the approximations.

More precisely, we present *upper bounds* for the cluster sizes (of clusters represented by the approximation) and these bounds are monotone decreasing with increasing dimensionality of the clusters/approximations.

Since a cluster (O, S) is a subset of an approximation (O_A, S), the cardinality $|O_A|$ is of course an upper bound for the size of O. Our aim is to develop tighter bounding functions $b((O_A, S))$. In other words, if we want to generate an actual cluster (O, S) based on the approximation (O_A, S) we can ensure that $|O| \leq b((O_A, S))$ in addition to $O \subseteq O_A$ holds. For the first bound we assume $\varepsilon_s < \frac{1}{|S|}$ (otherwise the naive bound $|O_A|$ still could be used for pruning).

For the definition of these bounds let us first provide some basic notions: Let $obj(O, S, x) \subseteq O$ be the set of objects with exactly x missing values in the dimensions S. Formally, $obj(O, S, x) = \{o \in O \mid |S \cap D_?(o)| = x\}$. And $greedy(O, S, y)$ the largest subset of O with in total y (or less) missing values (break ties arbitrarily). Formally, $greedy(O, S, y) = \text{argmax}_{M \subseteq O}\{|M| \mid \sum_{i=0}^{|S|} |obj(M, S, i)| \cdot i \leq y\}$. This set can efficiently be determined by a greedy approach, selecting the objects in increasing order of their missing values until y is exceeded.

Lemma 2 *Monotonicity of cluster sizes (ε_s)*
1) Given an mx-approximation $A = (O_A, S)$, the size of each valid cluster $C = (O, S)$ represented by the approximation is upper bounded by the function

$$b_s((O_A, S)) = |obj(O_A, S, 0)| \cdot \frac{1}{1 - |S| \cdot \varepsilon_s}$$

That is, $|O| \leq b_s((O_A, S))$ holds.
2) For each $S' \subseteq S$ the corresponding mx-approximation $B = (O_B, S')$ with $O_B \supseteq O_A$ fulfills: $b_s((O_B, S')) \geq b_s((O_A, S))$.

Proof 7.3 *1) In each dimension $\varepsilon_s \cdot |O|$ missing values are permitted; thus given the clustered objects O, at most $|S| \cdot \varepsilon_s \cdot |O|$ objects containing missing values can be selected because otherwise at least one dimension exceeds the threshold. By selecting objects without missing values we increase the potential for selecting objects with missing values. Thus the size of the clustered objects is in the best case given by $|O| = |obj(O_A, S, 0)| + |S| \cdot \varepsilon_s \cdot |O|$. Solving this equation for $|O|$ yields the desired upper bound.*
2) W.l.o.g. let $d \in S$ and $S' = S \backslash \{d\}$. The approximation O_A contains at most $\varepsilon_s \cdot |obj(O_A, S, 0)| \cdot \frac{1}{1-|S| \cdot \varepsilon_s} := k$ objects with missing values in dimension d; for the best

case we have to assume that these k objects have no missing values in the other dimensions, thus they are included in $obj(O_B, S', 0)$. Since obviously also $obj(O_B, S', 0) \supseteq obj(O_A, S, 0)$ holds, we get $|obj(O_B, S', 0)| \cdot \frac{1}{1-|S'|\cdot\varepsilon_s} \geq (|obj(O_A, S, 0)|+k) \cdot \frac{1}{1-(|S|-1)\cdot\varepsilon_s} = |obj(O_A, S, 0)| \cdot \frac{1}{1-|S|\cdot\varepsilon_s} \cdot \frac{2-|S|\cdot\varepsilon_s}{1+\varepsilon_s-|S|\cdot\varepsilon_s} \geq |obj(O_A, S, 0)| \cdot \frac{1}{1-|S|\cdot\varepsilon_s}$ □

Lemma 3 *Monotonicity of cluster sizes (ε_g)*
1) Given an mx-approximation $A = (O_A, S)$ with $|S| \leq mx$, the size of each valid cluster $C = (O, S)$ represented by the approximation is upper bounded by the function

$$b_g((O_A, S)) = |greedy(O_A, S, \varepsilon_g \cdot |O_A| \cdot mx)|$$

That is, $|O| \leq b_g((O_A, S))$ holds.
2) For each $S' \subseteq S$ the mx-approximation $B = (O_B, S')$ with $O_B \supseteq O_A$ fulfills: $b_g((O_B, S')) \geq b_g((O_A, S))$.

Proof 7.4 *1) In the best case O_A itself is a valid cluster; thus, the pattern tolerance would permit at most $\varepsilon_g \cdot |O_A| \cdot |S| \leq \varepsilon_g \cdot |O_A| \cdot mx$ missing values $\Rightarrow$ the size of the greedy set is an upper bound.*
2) Since $O_B \supseteq O_A$ and therefore $\varepsilon_g \cdot |O_B| \cdot mx \geq \varepsilon_g \cdot |O_A| \cdot mx$, the monotonicity holds obviously. □

Overall, also the relative thresholds enable to generate monotone behaving cluster approximations, and we can prove important bounds for the cluster sizes. In the following, these properties are used for our instantiation of the model but are of course not restricted to this instantiation. They enable in general an efficient processing for the constant threshold case as well as for the relative threshold case.

7.2.4 Discussion of parameters

In our model we introduce the three parameters ε_o, ε_s, and ε_g that correspond to the three different types of fault tolerances. These parameters are easy to interpret since they directly correspond to the fraction of missing values the user is willing to accept. By setting all parameters to 1, an arbitrary amount of missing values per clusters is allowed. However, setting at least one parameter to 0 leads to traditional subspace clustering where no missing values can occur in the clusters. In general, by increasing the parameters the tolerance w.r.t. missing values is eval-

uate less strict. Thus, more missing values per cluster are permitted and larger clusters can be obtained.

As mentioned in Section 7.2.2, none of the parameters subsumes any of the other parameters, i.e. they are all beneficial to specify the characteristics of a cluster. Though, they are not completely independent. Due to the object tolerance it always holds that $\sum_{o \in O} |S \cap D_?(o)| \leq \sum_{o \in O} \varepsilon_o \cdot |S| = |O| \cdot \varepsilon_o \cdot |S|$. Thus, the pattern tolerance cannot be violated if we select ε_g larger than ε_o. In such a case the pattern tolerance has no effect on the cluster. Accordingly, we get from the dimension tolerance $\sum_{o \in O} |S \cap D_?(o)| = \sum_{d \in S} |O \cap O_?(d)| \leq \sum_{d \in S} \varepsilon_s \cdot |O| = |S| \cdot \varepsilon_s \cdot |O|$. Thus, the user should select $\varepsilon_g \leq \min\{\varepsilon_o, \varepsilon_s\}$ to activate the pattern tolerance condition.

On the other hand, if one chooses $\varepsilon_o \geq \varepsilon_g \cdot |DB|$ it holds for all objects o of a cluster that $|S \cap D_?(o)| \leq \sum_{o \in O} |S \cap D_?(o)| \leq \varepsilon_g \cdot |O| \cdot |S| \leq \varepsilon_g \cdot |DB| \cdot |S| \leq \varepsilon_o \cdot |S|$, i.e. the object tolerance is trivially fulfilled due to the pattern tolerance. A similar property can be shown for the dimension tolerance. Overall, to use all three tolerances in a non-trivial way the condition $\max\{\frac{\varepsilon_o}{|DB|}, \frac{\varepsilon_s}{|Dim|}\} \leq \varepsilon_g \leq \min\{\varepsilon_o, \varepsilon_s\}$ has to hold.

7.3 Instantiation and FTSC Algorithm

In this section, we use the general model presented in Section 7.2 to extend traditional subspace clustering. To focus on the handling of missing values, we decided to keep the cluster definition as simple as possible. Further extensions to more complex clustering models are intended for future work. Our method is called FTSC (fault tolerant subspace clustering).

7.3.1 Subspace cluster definition

For our cluster definition we extend the widely used grid-based subspace clustering paradigm, e.g. used by CLIQUE [AGGR05] or SCHISM [SZ04]. The idea is to discretize the search space by partitioning the data space into non-overlapping rectangular grid-cells. Cells are obtained by partitioning every dimension into a given number of equal-length intervals and intersecting intervals from each dimension. A grid-cell is dense and, hence, a cluster if it contains more than a given number of objects. Assuming the database to be normalized between $0.0 - 1.0$ and using g intervals per dimension, a cluster $C = (O, S)$ is defined by

$\forall o, p \in O : \forall d \in S : \lfloor o_d \cdot g \rfloor = \lfloor p_d \cdot g \rfloor$, i.e. O contains the objects of a S-dimensional cell, and $|O| \geq minPts$, i.e. the grid cell is dense

To extend this approach to fault-tolerance, we additionally manage objects containing missing values. Thus, besides the objects within the same cell we have to consider for each dimension d the objects $O_?(d)$ to get the cluster approximations. If I^d denotes the objects of an one-dimensional interval for the dimension d, we have to consider the set $\bigcap_{d \in S} I^d \cup O_?(d)$ to obtain the approximation. However, keep in mind that Def. 7.6 restricts the maximal number of missing values allowed per object. Thus, some objects of $\bigcap_{d \in S} I^d \cup O_?(d)$ must not be included in the approximation. As an example, a cluster approximation for a cluster at cell $(2, 1)$ consists of the objects located in the cells $(2, 1)$, $(2, ?)$, $(?, 1)$ and $(?, ?)$. Assuming, e.g., at most one missing value is allowed per object we can immediately reject the cell $(?, ?)$. Furthermore, we use the redundancy model introduced in [AKMS08a] to exclude redundant clusters and to avoid an overwhelming result size.

7.3.2 Algorithmic processing

The general processing of FTSC is depicted in Algorithm 7.1. For generating the clusters we use a depth-first search through the subspace lattice (starting with low-dimensional subspaces going down to higher dimensional ones; line 11-14,20-22). Advantageous in using a depth-first search is the elimination of low-dimensional redundant clusters [SZ04, AKMS08a]: While *descending* the search tree we only generate the cluster approximations (line 5). The actual clusters are generated during the recursive *ascension* of the search tree; however, we only have to do this if the previously generated approximation is not already redundant to the identified clusters (line 15,16). Thus, in many cases we only generated the cluster approximations and not the clusters while traversing the subspaces.

For an efficient generation of cluster approximations, we use an enhanced representation of the corresponding object sets O_A: we perform a partitioning of this set according to the amount of missing values per object. An approximation (O_A, S) is represented by

$$(O_A, S) = ([O_A^0, O_A^1, \ldots, O_A^x], S)$$

where each partition $O_A^i \subseteq O_A$ ($i \in \{0, \ldots, x\}$) denotes the objects having exactly i missing values in S. Please note that x is upper bounded by Definition 7.6 due to the value of ε'_o, i.e. objects with too many missing must no be included in the

approximation. The advantage of this representation is the efficient calculation of approximations with increasing dimensionality as needed for a depth-first processing. Given the objects I of an one-dimensional interval for the dimension d and the objects $O_?(d)$ that exhibit a missing value in the same dimension (line 3-4), then the extension of the cluster approximation $([O_A^0, O_A^1, \ldots, O_A^x], S)$ to the higher-dimensional subspace $S' = S \cup \{d\}$ can be obtained by

- $O_B^0 = (O_A^0 \cap I)$
- $O_B^i = (O_A^i \cap I) \cup (O_A^{i-1} \cap O_?(d))$, $0 < i \leq x$
- $O_B^{x+1} = (O_A^x \cap O_?(d))$

generating $([O_B^0, \ldots, O_B^{x+1}], S')$. This efficient intersecting is possible due to the monotonicity of the object sets as proven in Lemma 1 and utilized in line 5. Furthermore, based on Definition 7.6 we can immediately prune the set O_B^{x+1} if $x+1$ is larger than the maximal number of missing values allowed per object (line 6). Since we used a valid approximation (O_A, S) for the generation of (O_B, S), we do not have to check this property for the sets O_B^i with $i \leq x$. Thus, this pruning can be done very efficiently since we solely have to check whether the set O_B^{x+1} can or can not be pruned.

Pruning of cluster approximations. To avoid analyzing all exponential many subspaces, our method uses the monotonicity properties introduced in Sec. 7.2.3 for pruning the search space. First, we exploit the monotonicity of the object sets itself since the approximations are obtained by intersecting with previously obtained ones (subset property; line 5).

Based on the monotonicity of the cluster sizes we can further increase the efficiency of FTSC. Since the cluster sizes are bounded by two different equations, we can reject an approximation if one of these bounds is smaller than $minPts$; we cannot get a valid cluster anymore. Because the bounds are monotone decreasing with increasing dimensionality, all approximations in higher-dimensional subspaces can also not yield valid clusters; we prune the whole subtree (line 8-10).

When using relative thresholds, the following improvement can be realized: At the beginning of the depth-first search we have to assume that valid clusters in the full-dimensional space exist, i.e. we potentially have to traverse the tree down to the subspace Dim. Since the mx-approximations are linked to the dimensionality of the clusters, we have to start with $mx = |Dim|$ to get a correct approximation. However, if the highest dimensional subspace was already processed or the sub-

Algorithm 7.1: The FTSC algorithm

function: $traverseSubspaces(.,.,.)$
input: previous approximation (O_A, S), current dimension d, maximal reachable dimensionality mx

```
1  S' ← S ∪ {d} // current subspace
2  foreach interval i of dimension d do
3      I ← getObjects(i,d) // objects of this interval
4      O?(d) ← getObjects(?,d) // objects with miss. vals. in d
5      (O_B, S') ← newApprox((O_A, S),I,O?(d)) // Lemma 1
6      prune objects with too many miss. vals. // Definition 7.6
7      if |O_B| < minPts then continue; // next interval
8      bound_s ← calc. bound of cluster size // Lemma 2
9      bound_g ← calc. bound of cluster size // Lemma 3
10     if bound_s < minPts ∨ bound_g < minPts then continue; // next interval
11     for dimension d' from d + 1 to |Dim| do // depth-first
12         // maximal dimensionality of reachable clusters
13         mx' ← |S'| + |Dim| − d' + 1
14         traverseSubspaces((O_B, S'), d', mx')
15     if (O_B, S') is not redundant to C ∈ Result then // cf. [AKMS08a]
16         (O, S') ← generateActualCluster((O_B, S'))
17         if |O| < minPts then continue; // next interval
18         if (O, S') is not redundant to C ∈ Result then // cf. [AKMS08a]
19             Result ← Result ∪ {(O, S')}

function: main()
20 for dimension d from 1 to |Dim| do
21     mx' ← |Dim| − d + 1
22     traverseSubspaces((DB, ∅), d, mx')
```

tree was pruned beforehand, we can refine the maximal realizable dimensionality. Thus, the value of mx can be decreased and we get tighter approximations of the clusters, which lead to better estimations for the sizes and hence more effective pruning (line 13,21). Overall, we use several pruning techniques to increase the efficiency of our FTSC algorithm.

Summary. We presented an instantiation of our fault tolerant model in grid-based subspace clustering; called FTSC. We used DFS along with powerful pruning techniques to search for cluster candidates. In the following, we compare FTSC to traditional subspace clustering that does not handle missing values.

7.4 Experimental Analysis

Setup. As there are no competing subspace clustering approaches that handle missing values, we compare FTSC with methods working on (complete) data, cleaned by statistical preprocessing techniques. In one case we use complete case analysis and in the second case mean value imputation. To ensure a fair comparison, we apply the grid-based clustering methods CLIQUE and SCHISM on these data since FTSC is also grid-based. The number of grid cells and the density parameters are identically chosen for each method. We generate synthetic data with in general 5000 objects, 30 dimensions and 10 clusters by the method used in [KKK04, AKMS08a]. Each cluster was hidden in 50-60% of the overall dimensions. As real world data we use data of the UCI archive [FA10] (pendigits, glass, breast cancer, vowel) and features extracted from sequence data (shape). We provide all datasets and executables on our website. We measure clustering quality via the $F1$ value [GFM$^+$11] frequently used in evaluation of subspace clustering [AKMS08a, MS08, MSE06]. . For real world data class labels are used as indicators for the natural grouping of objects. All experiments were conducted on Opteron 2.3GHz CPUs and Java6 64 bit.

Scalability. In our work we focus on the quality of FTSC; nonetheless, we briefly analyze the efficiency of the approach. In Figure 7.5 the dimensionality of the data, containing 5% missing values, is varied. All methods show an increase in their runtime. Not surprisingly, the runtime of FTSC is the highest since it has to analyze the complex missing values while the others simply work on pre-processed data. However, as we will analyze in the last experiment, our pruning methods effectively lower the runtimes and lead to an applicability of fault tolerance in

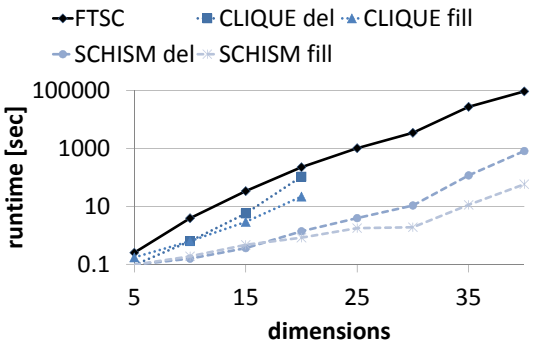


Figure 7.5: Runtime vs. dimensions

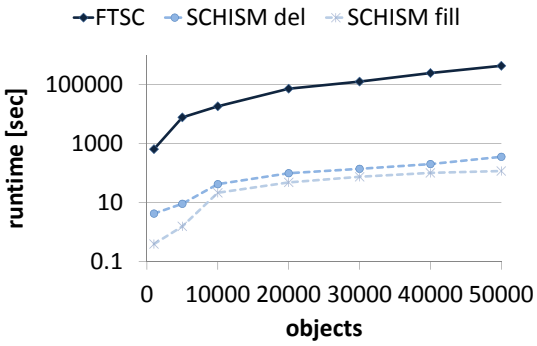


Figure 7.6: Runtime vs. database size

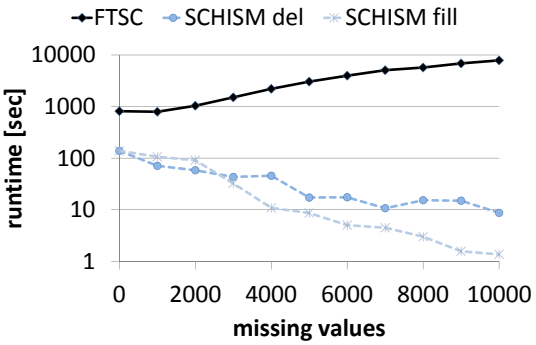


Figure 7.7: Runtime vs. number of missing values

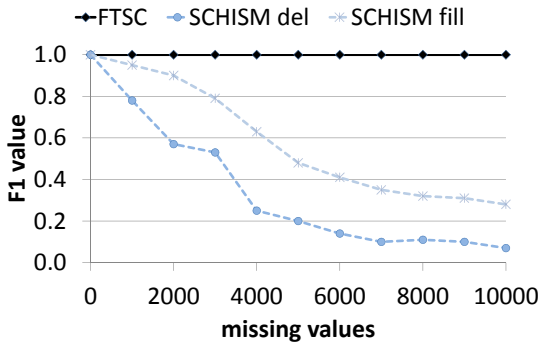


Figure 7.8: Missing values distributed randomly

practical applications. An important result is that CLIQUE is not applicable for high dimensionalities; the number of generated clusters is too large. Thus, in the remaining experiments we focus on FTSC and SCHISM. A similar trend can be observed in Figure 7.6 where the number of objects is varied. However, for this setting the increase in runtime is only moderate.

Most important is the runtime behavior with increasing number of missing values (Figure 7.7). The amount of missing values spread inside the dataset defines how much of the original inner structure is still present. The more missing values are spread, the less certain information is given. Therefore, we randomly add missing values to the data. FTSC shows a slight increase in runtime since with increasing number of missing values the cluster approximations get larger and pruning is more difficult. Overall, the runtime of FTSC is higher than SCHISM; however, this effort is worthwhile since it is accompanied by a high clustering quality as analyzed in the following experiments.

Clustering quality. In Figure 7.8 the same data as in Figure 7.7 is used. As shown, FTSC gets nearly perfect quality even with a high amount of missing values in the data. Our novel method based on relative thresholds countervails the errors in the data. For the setting with 0 missing values, the competing approaches get the same qualities. However, the quality of the delete method (complete cases analysis) drops heavily due to the high information loss. By filling up the missing values (single imputation), this effect can be slightly weakened; however, the errors that come along with the preprocessing eventually get too large, leading to poor quality. Overall, our FTSC method is the only choice for a flexible handling of missing values.

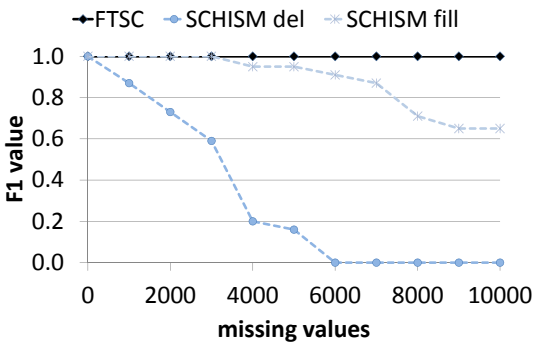


Figure 7.9: Missing values distributed outside clusters

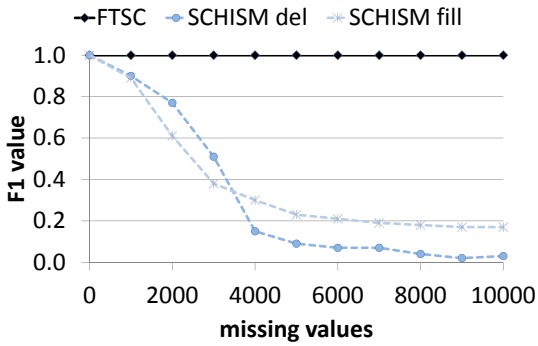


Figure 7.10: Missing values distributed inside clusters

While in Figure 7.8 the missing values are randomly distributed within the data, in Figure 7.9 we add them only to the dimensions of objects not covered by the hidden clusters. Thus, we do not obfuscate the hidden clusters with missing values. Again, our FTSC gets the highest quality while the delete approach drops heavily. Under this setting SCHISM-fill can keep up with FTSC until 4000 missing values. Since the missing values are distributed outside the clusters, they can be recovered; however, with a huge amount of missing values artificial clusters appear due to the preprocessing.

This observation is confirmed in Figure 7.10 where the data is generated the other way round: the missing values are distributed within the objects and dimensions of the hidden clusters. As expected, both SCHISM methods cannot handle this scenario; the clusters cannot be detected even with few missing values. Our FTSC is able to detect the hidden clusters with increasing number of miss. values.

Real world data. In the following experiments depicted in Figure 7.11-7.15 we analyze the clustering quality for real world data. We increase the number of randomly distributed missing values to analyze the methods' robustness to faults in the data. For all datasets the following observations become apparent: By adding 0 missing values, the qualities of all approaches are nearly identical. The small differences can be explained by slightly different clustering definitions. Our FTSC achieves the highest clustering qualities and shows robust behavior with increasing number of missing values. Even for a huge amount of missing values the quality is high and only for some datasets a small decrease can be observed. The methods based on pre-processing show a stronger decrease of their clustering qualities. Especially, the deletion methods are consistently worse than the methods based on

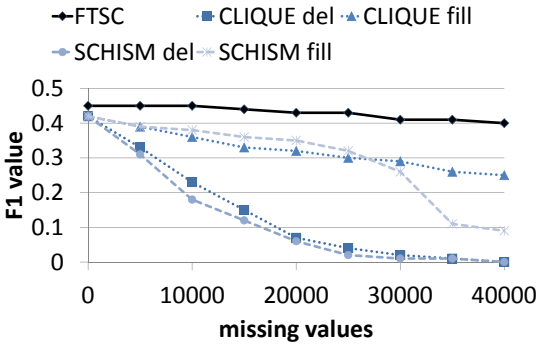


Figure 7.11: Quality on pendigits

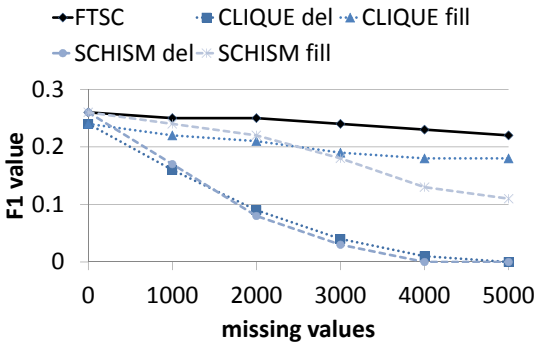


Figure 7.12: Quality on vowel

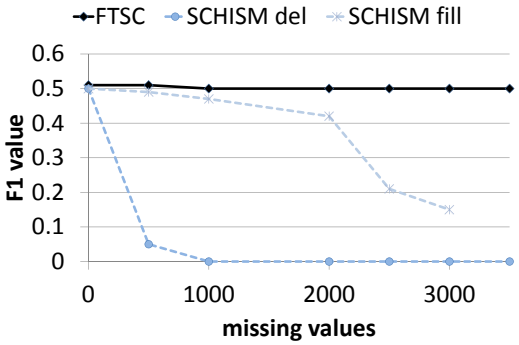


Figure 7.13: Quality on breast

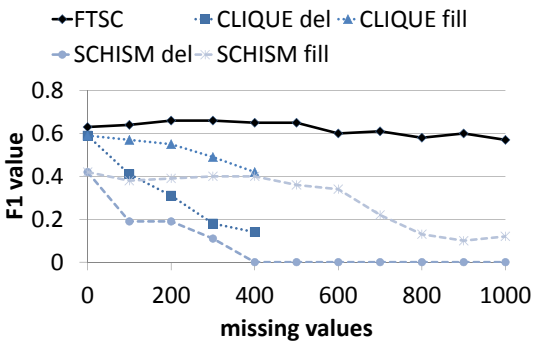


Figure 7.14: Quality on shape

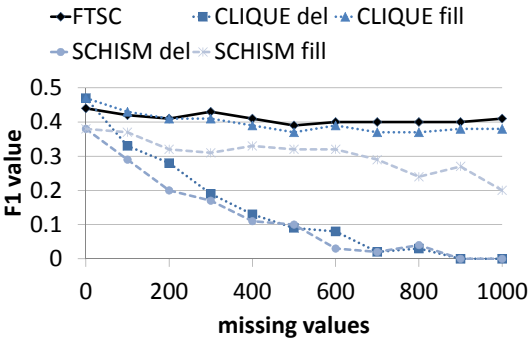


Figure 7.15: Quality on glass

data set (# miss. val)	*pendigits (10000)*	*vowel (3000)*	*breast (2000)*	*shape (400)*	*glass (400)*
FTSC	3340	187	203	1544	141
SCHISM del	313	1	109	3	47
SCHISM fill	797	140	156	16	15
CLIQUE del	593	94	-	405	1063
CLIQUE fill	1562	141	-	46	875

Figure 7.16: Runtime [msec] for real world data

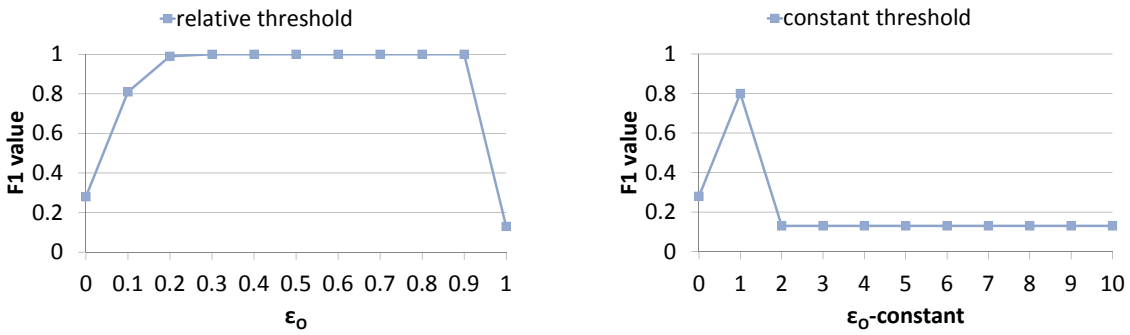


Figure 7.17: Relative vs. constant thresholds (clustering quality)

filling up missing values by mean value imputation. In Figure 7.16 the runtimes of the methods for an exemplary number of missing values is depicted. The efficiency of all approaches is high. Summarized, our FTSC shows good runtimes on real world data and gets highest clustering qualities even if the data is polluted by a huge amount of missing values.

Model analysis. At last, we analyze some characteristics of our approach. First we highlight the difference by using constant or relative thresholds in Figure 7.17. On the left we vary the relative threshold ε_o and determine the corresponding clustering qualities. As shown we identify all hidden clusters within the data and this can be realized by a wide range of parameter settings. The parameter is very robust and only for the extreme cases (near 0 or 1) the quality drops. On the other hand, by using constant thresholds (right figure, ε_o^c=absolute number of allowed missing values) we cannot get a perfect clustering. Moreover, a good clustering is only obtained for a single parameter setting. This experiment confirms that using constant thresholds is not meaningful for fault tolerant clustering approaches; instead our flexible fault tolerance should be used. Relative thresholds are very robust and are easily interpretable since they directly correspond to the fraction of missing values the user is willing to accept.

In Figure 7.18 we analyze the efficiency gain of our pruning methods on the pendigits data. We proved in Section 7.2.3 different monotonicities based on the three types of tolerances/thresholds. These properties are used in the FTSC algorithm for pruning the search space. In Figure 7.18 these pruning techniques are turned on or off. Consequently, the highest runtime is obtained by using no pruning, also setting up the baseline runtime (100% runtime; individually for the relative and constant case). Using just a single pruning method already yields a

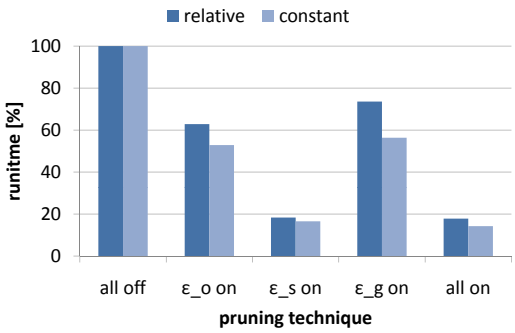


Figure 7.18: Effectiveness of pruning methods

decrease of the runtime. For this dataset the pruning based on ε_s performs best but for other data different results were obtained. However, by using all techniques simultaneously (rightmost bars), we always achieve the lowest runtime. On average, we achieve a runtime improvement of about 60-90% for all datasets. By using all the information derived in Section 7.2.3, our FTSC can efficiently determine a subspace clustering result even in the complex scenario of missing values.

7.5 Conclusion

In this chapter we proposed a method for handling missing values in the context of subspace clustering. We introduced a general fault tolerance model to handle missing values that can be applied to multiple subspace cluster definitions. Our integrated mining model handles missing values based on the currently considered subspace and set of objects, thus enabling objects to be part of multiple subspace clusters. With our variable thresholds we realize a flexible model that adapts to the individual cluster characteristics and that ensures a robust parameterization. The increased complexity of fault tolerant subspace clustering demands for effective pruning methods. Thus, we proved important monotonicity properties of our general model. As an instantiation of our general model we implemented a fault tolerant grid-based method. In the experimental evaluation, this approach yields high quality results on real and synthetic data even in the presence of many missing values.

While our FTSC method handles incomplete data as one type of imperfect information, in the next chapter we will present an integrated subspace clustering approach handling uncertain data.

Chapter 8

Subspace Clustering for Uncertain Data

Analyzing high dimensional uncertain data is a challenge in data mining research. While subspace clustering solves the problems of full-space clustering by analyzing subspace projections of the data, all existing methods rely on precise values. In scenarios where uncertain values occur, e.g. due to noisy sensor readings, one cannot expect that these algorithms deliver high-quality patterns. Especially, deciding whether dimensions are relevant for a subspace cluster is challenging in uncertain settings. Thus, approaches for effective subspace clustering on uncertain databases are needed.

In this chapter, we develop an integrated subspace clustering method handling uncertain data. The information provided by the individual distributions of objects is used in an effective manner. Because in uncertain scenarios a strict assignment of objects to single clusters is not appropriate, we enrich our model with the concept of membership degree. Subspace clustering for uncertain data is computationally expensive; thus, we propose an efficient algorithm. In thorough experiments we show the effectiveness and efficiency of our new subspace clustering method.

8.1 Motivation

Uncertainty is ubiquitous in application domains where data is processed [AY09]. Examples are sensor networks or data that is perturbated because of security rea-

sons. For databases, two definitions of uncertainty are coexistent: tuple uncertainty [DS07] and attribute uncertainty [CKP03]. While tuple uncertainty refers to whole database tuples, i.e. the probability of a tuple's validity, attribute uncertainty expresses imprecise information about the attribute values of a tuple, i.e. no exact values are given. Our model deals with attribute uncertainty.

In general, uncertain data can be divided into two categories: data with inherent uncertainty and data with artificially induced uncertainty. Data of the first category is attributable to imprecise real world measurements, for example due to noisy sensor readings, transmission inaccuracies, or hardware failures. A concrete example are local sensing applications such as GPS, where coordinates become invalid after very short time. In this case, exact tracking is infeasible. Another example are sensor networks, where sensor readings of different nodes can be inconsistent and a combination of these readings yields uncertainty. On the other hand, artificial uncertainty is intentionally added to the data. For example, before analyzing money transactions with a data mining method, the data is anonymized to ensure the users' privacy. Another example for artificial uncertainty is the information loss due to specific (technical) limitations; e.g. data is sampled, aggregated, or rounded to satisfy memory or bandwidth constraints.

Data mining on uncertain databases is critical since it is prone to errors in the data. That is, small value deviations can induce completely different results. Furthermore, attribute values with a large error are less reliable for data mining purposes than ones with small errors. Our novel subspace clustering methods considers these aspects to ensure robust clustering results. Since often uncertain objects are represented by probability density functions (*pdfs*), our subspace clustering methods analyses data objects modeled as (multi-dimensional) probability density functions.

Figure 8.1 recaps the idea of clustering in subspace projections of the data: C_1 denotes a full space cluster, i.e. both dimensions are relevant, while C_2 represents a cluster in the projection to dimension d_2. In d_1, the object values of C_2 are highly scattered, making d_1 irrelevant for C_2. This exclusion of dimensions is necessary to identify the clustering structure of the data. Since Figure 8.1 represents a data set with precise information, it can be handled by any existing subspace clustering method. The data illustrated in Figure 8.2, however, corresponds to an uncertain scenario. Uncertain objects are represented by percentiles of their distribution and for each dimension, the 5th, 25th, 75th, and 95th percentiles are given. None of

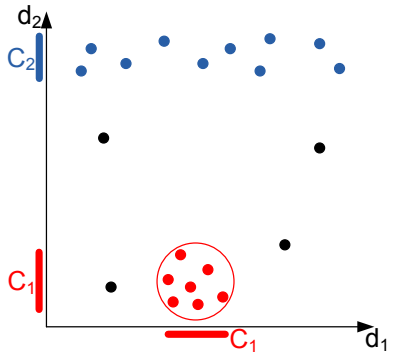


Figure 8.1: Exemplary subspace clustering for precise data

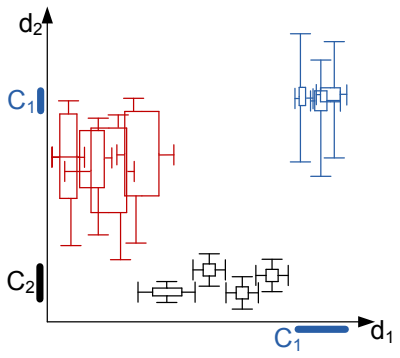


Figure 8.2: Challenge of subspace clustering for uncertain data

the existing subspace clustering methods can analyze such data. While there exist extensions of traditional full-space clustering algorithms to handle uncertainty, approaches for subspace clustering are still missing.

For subspace clustering on uncertain databases, several problems need to be solved. First, all clustering algorithms are based on object similarity. But how to measure similarity between uncertain objects? Simple solutions that remove uncertainty before using a standard clustering algorithm waste important information. By using an integrated mining method that does not rely on a preprocessing step the original data is analyzed and clustering quality can be increased.

Second, under uncertainty the decision whether a specific dimension is relevant for a subspace cluster is challenging. See again Figure 8.2 for an illustration. Comparing the groupings of blue and red objects, their individual uncertain objects have similar 5th and 95th percentiles, but the 25th and 75th percentiles differ significantly; i.e. for the red objects the values are distributed over a much higher range. Whether the groupings constitute clusters and what their relevant dimensions are cannot be answered easily. Intuitively, the blue objects correspond to a cluster with two relevant dimensions d_1,d_2: in d_1 the individual object values have overall a low dispersion and in d_2 the inner percentiles (25% , 75%) show that the distributions are concentrated on a short range in a similar position. In contrast, the inner percentiles of the red objects have a high extent in both dimensions. No cluster should be found here. A third case is illustrated by the black objects: for the individual objects the dispersion is low for both dimensions. In d_1, however, the objects themselves are highly scattered over the whole dimension, making d_1 irrelevant for cluster C_2.

And finally, the question of whether an object can belong to several clusters has to be solved. Partitioning clustering approaches that find disjoint clusters are obviously out of place in an uncertain setting. For instance, an uncertain object can belong to different clusters with the same probability.

In this chapter, we introduce an integrated mining approach for subspace clustering of uncertain data that tackles the issues mentioned above. To improve the clustering results, maximal usage of available information is achieved. We present our approach in three stages, which differ in their degree of information usage. Furthermore, besides presenting a partitioning version of our approach that may be desired by some applications, we introduce novel non-partitioning variants. They are also designed to make maximal usage of the information provided. Especially, we augment clusters with membership degrees of their assigned objects. This improves the quality of clusterings and enables users to extract more useful information out of them. Since our proposed model is computationally expensive, we present an efficient solution that uses Apriori-based pruning and heuristic sampling while still providing high quality results.

Summarized, our contributions are

- A novel integrated subspace clustering method for uncertain data.
- A novel approach to non-partitioning clustering based on membership degrees.
- An efficient algorithm that uses Apriori-based pruning and heuristic sampling.

Before presenting our model, the following section reviews the limitations of existing methods for subspace clustering of uncertain data.

8.2 Comparison with Related Work

As discussed in the first part of this thesis, performing full-space clustering might lead to questionable clustering results since irrelevant dimensions obfuscate the clustering structure. As a solution, subspace clustering [KKZ09, PHL04] was introduced to find clusters hidden in locally relevant subspace projections of the data. None of the existing subspace clustering approaches, however, can handle uncertain data.

Uncertainty is ubiquitous in several application domains, e.g. similarity search, indexing, data integration, skyline queries, stream processing, or data mining. A

thorough summary can be found in [AY09]. The simplest solution to handle such uncertain information is to clean the data in a pre-processing step, e.g. the *pdf* of each object is a-priori replaced by the expected value of the *pdf*. Consequently, after this transformation each object is represented by a usual numerical vector and any existing subspace clustering method can be applied. The drawbacks of such a pre-processing step were already discussed in Chapter 6 and 7 of this thesis: valuable information is no longer available and the characteristics of subspace clustering are not considered. Nonetheless, we include this variant as a baseline competitor for subspace clustering on uncertain data.

To circumvent the limitations of preprocessing, integrated clustering methods were introduced. The existing techniques, however, are restricted to full-space clustering. The methods of [CCKN06, CM08, NKC+06] adapt the k-Means principle to cluster uncertain data, while extensions of the density-based clustering paradigm were proposed by [KP05a, KP05b, XL08]. In contrast to these full-space clustering approaches, our novel model addresses subspace clustering for uncertain data.

Please note that our model for clustering uncertain data and traditional mixture models represent two orthogonal principles. Intuitively, in mixture models, each *cluster* (i.e. component) is represented by a probability distribution. The observed data, which is sampled from this distribution, however, corresponds to precise data, i.e. objects are represented by usual vector data. In our case, however, each *object* is described by an individual *pdf*. Also, our method should not be confused with kernel density estimation [Sil86]. Similar to mixture models, kernel density estimation techniques try to model the probability density function of the overall data; again, the actual objects are described by vector data.

8.3 Subspace Clustering for Uncertain Data

In this section, we present our model for clustering uncertain data in subspace projections of the data. As a preliminary, we introduce in Section 8.3.1 a general framework enabling the detection of subspace clusters. Based on this framework, we present our general subspace cluster definition in Section 8.3.2. We progressively increase the information usage obtained from the uncertain objects. In Section 8.3.3 we present how our overall clustering result is determined and how overlapping clusters are regarded.

8.3.1 Preliminaries: Subspace Clustering Framework

In the following, we describe a general framework to identify subspace clusters. This approach is similarly used, for example, in [PJAM02, YM03], and we adapt this principle to handle uncertain data. In general, a subspace cluster is a set of objects O along with a set of relevant dimensions S in which the objects group together. In the remaining non-relevant dimensions the objects show no good grouping.

Definition 8.1 *Subspace cluster*
Given a set of dimensions Dim and a database DB of uncertain objects within these dimensions, a subspace cluster C is defined by

$$C = (O, S) \text{ with } O \subseteq DB \text{ and } S \subseteq Dim$$

The crucial point is to determine which objects actually support a specific cluster in a specific subspace projection. Similar to the methods [PJAM02, YM03], we want to adapt the cell-based clustering paradigm. As shown in [MGAS09], approaches from this category are very efficient and they generate clusterings of high quality. Efficiency is important because analyzing *pdf*s is more complex than analyzing usual vector data. The basic idea of cell-based clustering (for precise data) is to approximate clusters via hypercubes. In the relevant dimensions of a subspace cluster the extent of the hypercube is limited by a maximal width w, while in the non-relevant dimensions the extent is unlimited. For example, the black objects in Figure 8.2 show a high extent in dimension d_1 while in d_2 the grouping is compact.

Obviously, each hypercube can be represented by a specific medoid. Given such a medoid m and a subspace S the support of a cluster around this medoid is calculated by

$$support_{certain}(m, S) = |\{o \in DB \mid d_\infty^S(m^S, o^S) \leq w\}|$$

where x^S is the projection of the point x to the corresponding subspace, and the distance is defined by:

Definition 8.2 *Chebyshev distance*
Given a subspace S and two $|S|$-dimensional points $x, y \in \mathbb{R}^{|S|}$, the Chebyshev distance for this subspace is defined by

$$d_\infty^S(x, y) = \max_{i=1,\ldots,|S|} \{|x_i - y_i|\}$$

Thus, all objects whose Chebyshev distance within the relevant subspace is smaller than the threshold w are included in the cluster.

For each subspace $S \subseteq Dim$ the support around the medoid m can be computed, and each cluster with a sufficiently high support ($\geq minSup$) is regarded as valid. Since, however, not all subspaces show a good clustering structure and most of them represent redundant information, one is only interested in the subspace S yielding the most interesting cluster (for a given medoid m). To measure the interestingness of a subspace cluster we introduce a quality function that accounts for the dimensionality of the cluster and its support. For this purpose we adapt the quality function of [PJAM02].

Definition 8.3 *Quality function*
Given a medoid m and a subspace S, let $support(m, S)$ be the support of the cluster C around m in the subspace S. The quality of C is defined by

$$quality(m, S) = support(m, S) \cdot (1/\beta)^{|S|}$$

where $\beta \in (0, 1]$ permits to trade off the cluster's support against its dimensionality.

Algorithm 8.1: General framework for subspace clustering

```
repeat
    bestQuality = −∞; bestMedoid = null; bestSubspace = null
    medoids = 0
    while medoids < numMeds do
        m = selectNextMedoid()
        medoids = medoids + 1
        foreach S ⊆ Dim do
            if support(m, S) < minSup then continue;
            if quality(m, S) > bestQuality then
                bestQuality = quality(m, S)
                bestMedoid = m; bestSubspace = S
    generateCluster(bestMedoid, bestSubspace)
    updateMedoidSelectionStrategy()
until no more clusters identified
```

Using this definition of support and quality, the general framework to find a subspace clustering is presented in Algorithm 8.1. Similar to the Monte Carlo methods [PJAM02, YM03], one *iteratively* tries to identify *single* good subspace

clusters within the database (lines 2-12). This is accomplished by randomly selecting uncertain objects out of the database that should represent the medoids of different clusters (line 5). For each of these medoids we identify the subspace for which the corresponding subspace cluster around the medoid results in the highest quality (line 7-11). Thus, the subspace with the most interesting clustering structure is selected for this medoid. In the end, we will have identified the medoid that is the best among all randomly selected medoids and we generate the corresponding final subspace cluster (line 12).

After generating a single cluster, we have to update the medoid selection strategy to avoid generating similar clusters several times (line 13). For precise data, the methods of [PJAM02, YM03] use a simple *partitioning* approach to cope with the problem of redundant clusters. If a subspace cluster is identified, the corresponding objects are removed from the set of potential medoids. Additionally, these methods also remove the objects from the database. This way, clusters detected in subsequent iterations obviously represent different sets of objects; the clusters are disjoint. The termination of this approach is guaranteed because the minimal required size/support of a cluster is fixed to a constant value and in each iteration one removes objects from the database. In Section 8.3.3, we propose our novel update strategies for uncertain data.

Overall, as we can seen, the support of a cluster around the medoid is the most important aspect that we need to define to detect meaningful subspace clusters. For traditional, i.e. certain, data the support can easily be determined by simply counting the number of objects that are near to the medoid: it is a binary decision whether an object exceeds the maximal distance or not. For uncertain data defining the clusters' support is more complex.

8.3.2 Support of Subspace Clusters in Uncertain Data

In this section we introduce our subspace cluster definition for uncertain data. In contrast to the traditional subspace clustering task, where each object is represented by a single high-dimensional vector, we have to consider uncertain objects. In our general approach each uncertain object is described by a probability density function (*pdf*). The *pdf* indicates the likelihood for an object to be located in a specified region. For a database DB with n uncertain objects in a d-dimensional space, we have n *pdfs* $p_i(x)$ with $x \in \mathbb{R}^d$ and for all $i = 1, \ldots, n$ the following

condition is fulfilled:

$$\int_{x \in \mathbb{R}^d} p_i(x) = 1$$

The terms object and *pdf* of an object are used interchangeably in this work. If we need to address the components of a vector x, we use the notation $p_i(x_1, \ldots, x_d)$ instead of $p_i(x)$.

Subspace clustering analyzes the clustering of objects in projections of the data space. For precise data individual components of the object vectors are skipped to obtain the projections. In our case, however, we have to consider the *pdfs* of uncertain objects. Formally, the projection of an uncertain object p_i to a subspace S is defined by:

Definition 8.4 *Projection of an uncertain object*
Given a pdf p_i and a subspace $S \subseteq Dim = \{1, \ldots, d\}$, the projection of p_i to S is the marginal distribution *of p_i for S. The obtained pdf is called*

$$p_i^S(x) \text{ with } x \in \mathbb{R}^{|S|}$$

For example and w.l.o.g. $S = \{1, \ldots, s\}$, then

$$p_i^S(x) = p_i^S(x_1, \ldots, x_s) = \int_{x_{s+1}, \ldots, x_d \in \mathbb{R}} \cdots \int p_i(x_1, \ldots, x_d)$$

i.e., we marginalize over the dimensions $\{s+1, \ldots, d\}$.

After defining the objects and their projections, we now introduce our subspace cluster definition. As mentioned above, we want to adapt the cell-based clustering paradigm. Please note, however, that in our case the shape of a cluster is actually not restricted by a hypercube of with w because we consider *pdfs*. This is even true if the *pdf*s' support is finite. Informally, in our model we require that the uncertain objects are very likely to be located within the corresponding hypercube.

The question in the following is how to determine the support of a subspace cluster for uncertain objects. We want to use as much information as possible to determine the quality of a cluster (based on the support, cf. Definition 8.3). Additionally, we have to consider which objects belong to the final cluster. For this purpose, it has to be discussed if a binary decision as in the certain case is useful.

Expectation based support. In a first naive approach each *pdf* is transformed to a single vector that represents the uncertain object and that can therefore be used to calculate the support of a cluster. This step corresponds to the pre-processing as mentioned in Section 8.2. A common representative is the expected position $E(p_i, S) \in \mathbb{R}^{|S|}$ of the object p_i in the subspace S. This expectation value is calculated via

$$E(p_i, S) = \int_{x \in \mathbb{R}^{|S|}} x \cdot p_i^S(x)\, dx$$

With these representations we can directly use our distance function to measure if an object is near the medoid, i.e. if the distance is smaller than w. Thus, the support of a cluster around the medoid m in the subspace S is

$$support_{exp}(m, S) = |\{p_i \in DB \mid d_\infty^S(E(m, S), E(p_i, S)) \leq w\}| \tag{8.1}$$

As in the certain case, it is easy to decide which objects belong to the final cluster because it is a binary decision whether p_i contributes to the support or not.

Besides being simple, this first approach is not sufficient to cluster uncertain objects. Let us consider the uncertain objects in the left region of Figure 8.2. In dimension d_2 the expected positions of the objects do not scatter and thus a relevant dimension is assumed. The uncertain objects, however, show a high variance in this dimension and hence the dimension is not a good candidate for clustering; that is, non-relevant dimensions are wrongly identified as relevant by this first approach. At the same time it is possible to miss some relevant dimensions of a subspace cluster, especially if skewed or multi-modal distributions are present. Similar problems can occur if we use other expectation based methods.

Specifically for subspace clustering we have to solve these problems. If we did not, we would generate low quality groupings, and we would identify the wrong relevant dimensions for our subspace cluster. Misinterpretation of the clustering result is possible.

Minimal probability based support. In our next method, we do not transform the uncertain objects to (certain) vector data but we exploit the information given by the *pdf*s more accurately. We do not calculate an actual distance value but we calculate the probability that two objects are near to each other. Formally, the probability that the distance between two independent objects p_i and p_j in a

subspace S is smaller than a maximal distance w is

$$P_{\leq w}(p_i, p_j, S) = \int_{\substack{x,y \in \mathbb{R}^{|S|} \\ d_\infty^S(x,y) \leq w}} p_i^S(x) \cdot p_j^S(y)\, dx\, dy \tag{8.2}$$

We have to integrate over all possible vectors whose distance to each other is small enough and multiply the corresponding densities of the underlying *pdfs*.

Please do not confuse this probability with the values computed, for example, in the traditional EM-clustering based on Gaussian mixture models. In mixture models, we compute for each object the likelihood of belonging to the cluster, i.e. we evaluate the density of a *single pdf* (the cluster's component) at a *given* realization (the observed data point). Here, we compute the probability that *any two* realizations of the *two given pdf*s are close to each other.

The support of a subspace cluster is now determined by all objects whose probability for the above event is high enough. Thus, besides the *maximal* distance w we introduce the *minimal* probability $minProb$ that an object has to exceed in order to increase the support. Overall, we define the support of a cluster in subspace S around the medoid m by

$$support_{minProb}(m, S) = |\{p_i \in DB \mid P_{\leq w}(m, p_i, S) \geq minProb\}| \tag{8.3}$$

Again we have a binary decision which objects belong to the cluster. In this new variant, however, the relevant dimensions of a cluster are better detected. For example, in Figure 8.2 the dimension d_2 is not selected as relevant for the red objects: due to the large scatter the probability for a small distance is not high enough.

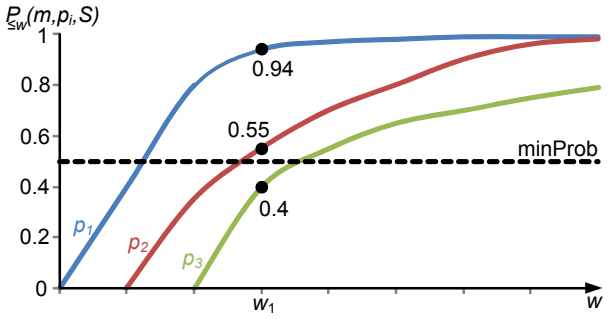


Figure 8.3: Probability distribution with varying w

Problems of this second method are presented in Figure 8.3. Given a fixed medoid m, the maximal distance w is plotted on the x-axis. The y-axis indicates the probability that an object is nearer to m than the current w. Three different objects are analyzed in the figure. Given the distance threshold w_1 and a minimal probability of $minProb = 0.5$, the object p_3 does not contribute to the support of the cluster. If many objects marginally miss the threshold, the calculated support will be underrated. The probability threshold is a hard bound for the objects to be included in the cluster.

Furthermore, the two other objects increase the support of the cluster because the probability is above the required threshold. Both objects, however, are considered equally important for the cluster, i.e. each object increases the support by 1, even if their probability values differ. In reality, the object p_1 is more likely clustered with the medoid than the object p_2. Thus, the $minProb$-based method does not distinguish between clusters that with a high probability occur in the database and clusters that are improbable. This is related to the fact that dense clusters are usually preferred over sparse clusters. In Figure 8.4, both clusters would get the same support value with this method; the cluster on the right should be preferred because the uncertainty within this clustering structure is very low.

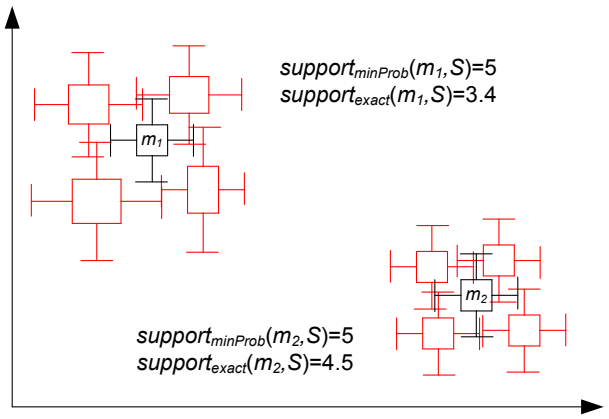


Figure 8.4: Minimal probability vs. exact probability

Exact probability based support. In this last approach we prefer clusters that appear with a high probability in the feature space. Until now each object in a cluster has the same weight, i.e. importance, for that cluster if the thresholds are exceeded (binary decision). In our following idea we use the probabilities

in a more advanced way by weighting the objects according to their likelihood; accordingly, a possibility to calculate the support is $\sum_{p_i \in DB} P_{\leq w}(m, p_i, S)$ for given m and S. In contrast to $support_{minProb}$, objects with a higher probability increase the support and thus the quality of the cluster to a higher degree. At the same time, we do not exclude objects via a hard bound of a threshold; that is, we can separate clusters based on the magnitude of their supporting objects and not only based on their size. With this support definition, the cluster on the right in Figure 8.4 has a higher support than the other cluster. An interesting aspect is that the support can be any positive real number and is not restricted to natural numbers. Hence a more fine-grained indication of the support is possible.

The definition of the support, as proposed above, includes all objects, even if their probability to be near the medoid is very low. These objects might be undesirable for the user and might be removed from the cluster. To optionally prohibit objects with a probability near to zero, we include a probability constraint $\varepsilon Prob$ that each object has to exceed. Thus, our final support definition is

$$support_{exact}(m, S) = \sum_{\substack{p_i \in DB \\ P_{\leq w}(m, p_i, S) \geq \varepsilon Prob}} P_{\leq w}(m, p_i, S) \tag{8.4}$$

As before, we can calculate the support and the quality of a subspace cluster with this formula. Thus, in Algorithm 8.1 we can determine the best medoid m and the best subspace S for which we generate the surrounding cluster. Our novel support criterion, however, no longer corresponds to a binary decision whether an object is included in the cluster or not. What is the final cluster presented to the user? Obviously, we could select all objects that account to the sum but this would contradict our assumption that each object has a different weight for the support.

We extend the notion of subspace cluster to subspace cluster with membership degree. Each object within a cluster is associated to a membership degree that indicates the importance of the object for the cluster or the degree of this object belonging to the cluster.

Definition 8.5 *Subspace cluster with membership degree*
Given a set of dimensions Dim and a database DB of uncertain objects within these dimensions, a subspace cluster C with membership degree is defined by

$$C = (O, S, deg) \text{ with } O \subseteq DB \text{ and } S \subseteq Dim$$

and $deg : O \rightarrow (0, 1]$ is a function that maps each object of the cluster to a specific membership degree.

For our instantiation we use: Let m be the medoid of a hypercube and S the current subspace, the membership degree is given by $deg(p) = P_{\leq_w}(m, p, S)$ for $p \in O$.

Thus, based on the support function $support_{exact}$, the cluster around the medoid m in subspace S is given by $C = (O, S, deg)$ with $O = \{p_i \in DB \mid P_{\leq_w}(m, p_i, S) \geq \varepsilon Prob\}$, and the membership degree exactly represents the probability that the object p is located in the hypercube around the uncertain object m.

Please note that the membership degree is computed with respect to a single cluster. Thus, we are not allowed to infer information in comparison to other clusters. Especially, the membership degree of an object p_i can exceed the value 1 summed over several clusters. E.g., for two clusters C_1 and C_2 the degree of object p_i can be 0.6 and 0.7. High membership degrees of a single object for several clusters are possible because in subspace clustering objects can belong to several different clusters due to the individual subspaces.

8.3.3 Determining the overall clustering.

In the previous sections we described how a single (best) cluster is found in an uncertain database using different support definitions. Our overall goal, however, is to identify a set of clusters that describe the characteristics of the whole dataset. The obtained clusters should be discriminable in the sense that the same clustering structure is not identified twice, i.e. redundant information should be avoided. To achieve this aim, we update the medoid selection strategy (Alg. 8.1, line 13).

As mentioned, existing methods [PJAM02, YM03] perform a *partitioning* of the objects since the already clustered objects are removed from the set of medoids as well as from the database. Formally, given the set of clusters $\{C_i = (O_i, S_i) \mid i = 1, \ldots, k\}$ generated so far in k iterations of the inner part of Algorithm 8.1, they draw the medoids in the $(k + 1)$th iteration uniformly and randomly from

$$candMedoids_{k+1} = DB - \bigcup_{i=1}^{k} O_i$$

That is, the objects already covered by clusters are removed from the database and, consequently, are not used as medoids anymore. To ensure the partitioning, the

clustered objects are also restricted to these subsets. Thus, no redundant clusters can be generated. Obviously, in the first iteration we have $candMedoids_1 = DB$. According to [YM03], we select $numMeds = 2 \cdot |candMedoids_{k+1}|/minSup$.

An uncertain object contributes to a cluster only in a specific amount. In a partitioning approach, already clustered objects cannot support another cluster. Thus, it seems reasonable to handle uncertain objects in a different way. Furthermore, in subspace clustering a non-partitioning approach can lead to better groupings because some objects might be clustered in several different dimensions. For these reasons, we introduce two novel non-partitioning clustering methods. These methods are based on our cluster selection procedure but can also be transfered to other subspace cluster definitions.

In our *first non-partitioning* approach we do not remove the objects from the database, i.e. the clustered objects are not restricted to the above defined candidate subset; all objects can support a new cluster even if they are already contained in several other clusters. The set of possible medoids, however, is reduced in each iteration until this set is empty or no more clusters are identified in the inner part of Algorithm 8.1. Thus, the termination of the method is again guaranteed.

This first non-partitioning variant is appropriate for our $minProb$-based support definition because this method adds only objects with high probability values to the clusters. However, our exact support definition is able to include further objects also with smaller probabilities; all these objects are excluded from the candidate set for the medoid selection in following iterations. Thus, this procedure might not be meaningful for the exact support definition since several clusters might be missed.

Consequently, we introduce our *second variant of non-partitioning* subspace clustering for the support definition $support_{exact}$. This variant uses the additional knowledge about the membership degree of the clustered objects. The basic idea is that an object with a small membership degree for the already known clusters is possibly included in further clusters. Thus, it is reasonable to use this object as a medoid in later iterations. However, the probability of this object to be selected is small compared to those objects that are not included in any clusters; the probability is high compared to objects that are in several clusters or to objects that are very likely in clusters. Consequently, we use a weighted sampling to determine our medoids in each iteration. Let $\{C_i = (O_i, S_i, deg_i) \mid i = 1, \ldots, k\}$ be the set of subspace clusters with membership degree detected in the iterations 1 to k.

The medoids in iteration $k+1$ of Algorithm 8.1 are drawn from DB based on a weighted sampling with the weights

$$\forall o \in DB: \ weight(o) = \begin{cases} 1, \text{ if } o \in (DB - \bigcup_{i=1}^{k} O_i) \\ \max\{0, 1 - \sum\limits_{\substack{i=1 \\ o \in O_i}}^{k} deg_i(o)\}, \text{ else} \end{cases}$$

Initially each object has the weight 1 and hence we draw the medoids uniformly out of the whole database. The procedure terminates because the weights are continuously decreasing. Especially an uncertain object that has been used as a medoid for cluster generation can never be selected twice: its membership degree must be 1 and hence its weights in following iterations are 0. The number of medoids $numMeds$ selected in the current iteration is determined by $2 \cdot \sum_{o \in DB} weight(o)/minSup$, i.e. the cardinality used before is replaced by the sum of weights. With this method we can generate overlapping clusters based on the membership degree of the uncertain objects.

Summary Overall, in our model we can distinguish between three different cluster definitions that vary in the calculation of the clusters' support. Furthermore, we introduce different methods to identify the overall clustering and we consider the overlap of clusters by our non-partitioning approaches. Table 8.1 gives an overview of the possible combinations and their corresponding names as used in the experiments.

	Partitioning	Non-Partitioning
$support_{exp}$	Exp-P	Exp-NP
$support_{minProb}$	MinProb-P	MinProb-NP
$support_{exact}$	Exact-P	Exact-NP

Table 8.1: Overview of our algorithms

8.3.4 Discussion of Parameters

Our model can be adapted based on different parameters. Most of the parameters, however, were already introduced in existing subspace clustering methods. To specify the values of $numMeds$, $minSup$, β, and w we refer to the guidelines

and heuristics proposed in [PJAM02, YM03, YM05]. According to [KP05a], we recommend for the parameter $minProb$ a value of 0.5. In this case, the support of a cluster is determined by all objects whose likelihood of being in the neighborhood of the medoid is larger than their likelihood of being far away, i.e. $P_{\leq_w}(m, p_i, S) \geq 1 - P_{\leq_w}(m, p_i, S)$ has to hold. Finally, we recommend the choice $\varepsilon Prob \approx 0.12$, which is motivated by the following observation: Consider a medoid m whose *pdf* is normally distributed according to $\mathcal{N}(0, w)$. Intuitively, if we sample from m's distribution, most of the samples p should also belong to the cluster of m. As a rule of thumb, objects located outside the 99th percentile of a distribution are often regarded as outliers. Thus, given $P(-\theta \leq X \leq \theta) = 0.99$ with $X \sim \mathcal{N}(0, w)$, samples in the range $[-\theta, \theta]$ should belong to the cluster. If we now consider the sample p at position θ itself as an uncertain object with *pdf* $\mathcal{N}(\theta, w)$, we get $P_{\leq_w}(m, p, S) \approx 0.12$. Thus, by selecting $\varepsilon Prob \approx 0.12$, the object p belongs to the cluster centered at m.

8.4 Algorithmic Properties

In the following section we discuss some algorithmic properties to ensure an efficient execution of our approach. The most important aspect in Algorithm 8.1 is the determination of the best subspace for a selected medoid, i.e. the subspace that yields the highest quality. In order to do so, we have to calculate the support of the current subspace cluster.

Anti-monotonicity property. Two of our techniques exploit an anti-monotonicity property that is valid for our subspace clustering model. Let O_S, O_T be the set of objects that supports the cluster around the medoid m in subspace S and T respectively; then for all subspaces $T \supseteq S$ we have $O_T \subseteq O_S$. Similar: $support(m, T) \leq support(m, S)\ \forall T \supseteq S$.

In this paragraph we prove that this property holds for all three variants of our support definition, starting with the expectation based variant.

Lemma 4 *Given a medoid m and subspaces T, S with $T \supseteq S$, for the expectation based support it holds: $O_T \subseteq O_S$ and $support_{exp}(m, T) \leq support_{exp}(m, S)$*

Proof 8.1 *W.l.o.g. we assume $S = \{1, \ldots, s\}$ and $T = S \cup \{s + 1\}$. For two points*

$x, y \in \mathbb{R}^{|T|}$ and their projections to S, denoted as $x^S, y^S \in \mathbb{R}^{|S|}$, it holds that

$$d_\infty^S(x^S, y^S) = \max_{i=1,\ldots,s}\{|x_i - y_i|\} \leq \max_{i=1,\ldots,s,s+1}\{|x_i - y_i|\} = d_\infty^T(x, y)$$

and

$$d_\infty^T(x, y) \leq w \Rightarrow d_\infty^S(x^S, y^S) \leq w$$

Using for x, y, x^S, and y^S the required expectation values of the pdfs we get

$$O_T \overset{(8.1)}{=} \{p_i \in DB \mid d_\infty^T(E(m,T), E(p_i,T)) \leq w\}$$
$$\subseteq \{p_i \in DB \mid d_\infty^S(E(m,S), E(p_i,S)) \leq w\} \overset{(8.1)}{=} O_S$$

Accordingly, $support_{exp}(m,T) = |O_T| \leq |O_S| = support_{exp}(m,S)$. □

The remaining two support definitions do not use the distance function directly but the probability $P_{\leq w}(p_i, p_j, S)$. Therefore, we first prove the following property:

Lemma 5 *Given two uncertain objects p_i, p_j and subspace T, S with $T \supseteq S$, it holds: $P_{\leq w}(p_i, p_j, T) \leq P_{\leq w}(p_i, p_j, S)$*

Proof 8.2 *W.l.o.g. we assume $S = \{1, \ldots, s\}$ and $T = S \cup \{s+1\}$. Then, for $x, y \in \mathbb{R}^{|T|}$ and their projections to S $x^S, y^S \in \mathbb{R}^{|S|}$, it holds*

$$\begin{aligned}
P_{\leq w}(p_i, p_j, T) &= \int_{\substack{x,y \in \mathbb{R}^{|T|} \\ d_\infty^T(x,y) \leq w}} p_i^T(x) \cdot p_j^T(y)\, dx\, dy \\
&= \int_{\substack{x,y \in \mathbb{R}^{|T|} \\ \max_{i=1,\ldots,s+1}\{|x_i - y_i|\} \leq w}} p_i^T(x) \cdot p_j^T(y)\, dx\, dy \\
&= \int_{\substack{x^S, y^S \in \mathbb{R}^{|S|} \\ \max_{i=1,\ldots,s}\{|x_i - y_i|\} \leq w}} \int_{\substack{x_{s+1}, y_{s+1} \in \mathbb{R} \\ |x_{s+1} - y_{s+1}| \leq w}} p_i^T(x) \cdot p_j^T(y)\, dx\, dy \\
&\leq \int_{\substack{x^S, y^S \in \mathbb{R}^{|S|} \\ d_\infty^S(x^S, y^S) \leq w}} \int_{x_{s+1} \in \mathbb{R}} \int_{y_{s+1} \in \mathbb{R}} p_i^T(x) \cdot p_j^T(y)\, dx\, dy \\
&= \int_{\substack{x^S, y^S \in \mathbb{R}^{|S|} \\ d_\infty^S(x^S, y^S) \leq w}} p_i^S(x^S) \cdot p_j^S(y^S)\, dx^S\, dy^S \\
&= P_{\leq w}(p_i, p_j, S)
\end{aligned}$$

□

Using this property, we now show:

Lemma 6 *Given a medoid m and subspaces T, S with $T \supseteq S$, for the minimal prob. based support it holds: $O_T \subseteq O_S$ and $support_{minProb}(m,T) \leq support_{minProb}(m,S)$*

Proof 8.3 *Based on Lemma 5 we get $P_{\leq_w}(p_i,p_j,T) \geq const \Rightarrow P_{\leq_w}(p_i,p_j,S) \geq const$. Thus, $O_T \overset{(8.3)}{=} \{p_i \in DB \mid P_{\leq_w}(m,p_i,T) \geq const\} \subseteq \{p_i \in DB \mid P_{\leq_w}(m,p_i,S) \geq const\} \overset{(8.3)}{=} O_S$. By choosing $const = minProb$ and $support_{minProb}(m,X) = |O_X|$ we get the required result.* □

Lemma 7 *Given a medoid m and subspaces T, S with $T \supseteq S$, for the exact probability based support it holds: $O_T \subseteq O_S$ and $support_{exact}(m,T) \leq support_{exact}(m,S)$*

Proof 8.4 *Similar as in Proof 8.3 but with $const = \varepsilon Prob$, and using the result $P_{\leq_w}(m,p_i,T) \leq P_{\leq_w}(m,p_i,S)$ in the summation as required for the definition of $support_{exact}$.* □

Candidate generation and pruning. Based on the anti-monotonicity property we can use the Apriori principle known from association rule mining [AS94] to enhance the performance of our algorithm. We efficiently generate and prune cluster candidates of higher dimensionality based on the lower dimensional projections for one medoid.

We start with the 1-dimensional subspaces and calculate the support of the resulting clusters around a medoid. All subspaces that do not exceed the $minSup$ threshold are discarded (cf. Alg. 8.1). Afterwards we can obtain candidates for the $(x+1)$-dimensional subspace clusters based on the retained x-dimensional ones. This procedure stops if the $minSup$ threshold is not exceeded by any of the generated candidates. With the Apriori method we prevent to analyze all exponential many subspaces for one medoid (line 7 of Algorithm 8.1) by early pruning of several subspaces.

Pruning based on quality estimation. While our first method prunes candidates w.r.t. a single medoid, our second approach makes use of already detected clusters, which could also be obtained from already processed medoids. We estimate the quality of all possible higher-dimensional projections of a cluster yet

to come. We know the maximal support (because it can only decrease, cf. anti-monotonicity) and the maximal dimensionality (union of the retained dimensions) and hence an upper bound for the quality can be determined. This allows an additional pruning if the quality of the currently best found cluster, potentially based on another medoid, is already above this upper bound. In this case, we do not need to generate the high-dimensional projections.

Sampling method. In some scenarios it might be possible to compute the exact values for $P_{\leq w}(m, p_i, S)$ by solving the integral in Equation (8.2). Assuming any possible *pdf*, however, this is an infeasible task. In our implementation we use Monte-Carlo sampling to approximately calculate the integrals and hence we realize an efficient method.

Each *pdf* p_i is represented by m objects $I = \{o_i^1, o_i^2, \ldots, o_i^m\}$ drawn from the distribution obtained by p_i. Given two *pdfs* p_i and p_j and their sampled objects I and J respectively, we denote the projections of the objects in I and J to the subspace S with I^S and J^S. Then we can approximate $P_{\leq w}(p_i, p_j, S)$ by

$$\frac{1}{|I^S| \cdot |J^S|} \cdot |\{(a, b) \in I^S \times J^S \mid d_\infty^S(a, b) \leq w\}|$$

That is, we compute the pairwise distances between the sampled objects and use the percentage of the objects pairs that have a smaller distance than w.

Summarized, we use three techniques to increase the efficiency of our algorithm. The Apriori principle is used to compute the set of uncertain objects that support the cluster in the current subspace. With the quality estimation we are able to prune further subspaces based on already detected clusters. Finally, the efficient sampling approach approximates the expensive integral computation.

8.5 Experimental Analysis

8.5.1 Setup.

Because there exist no direct competitors in the domain of subspace clustering for uncertain data, we compare our approaches with UK-Means [CCKN06] and Proclus [AWY+99]. UK-Means is chosen as a representative for fullspace clustering on *uncertain* data while Proclus identifies *subspace* clusters on certain data. The number of clusters k, required for both algorithms, is set to the true number of

clusters hidden in the data. The average dimensionality for Proclus is also obtained by the hidden clusters. According to [YM03], we use $\beta = 0.25$ and we calculate the minimal support based on the database size, i.e. $minSup = \alpha \cdot |DB|$ with $\alpha = 0.07$. Furthermore, $w = 6$, $minProb = 0.5$, and $\varepsilon Prob = 0.1$.

We generate synthetic data by adapting the methods from [AWY+99, PJAM02, YM03] to obtain overlapping clusters with uncertain objects. From each probability density function 10 samples are drawn. Unless stated otherwise, we generate data with 16 dimensions, 1000 objects, 10 clusters, an average cluster dimensionality of 25% from all dimensions, 15% overlapping objects, and 10% uniformly distributed noise objects. The *pdfs* are Gaussian distributions with a variance of 1% of the data range in the relevant dimensions and up to 50% of the data range in the non-relevant ones. To study the performance on real world datasets we use data that is available at the UCI archive [FA10] (pendigits, glass, breast cancer). In addition, we use features extracted from sequence data in [KWX+06] (shape). These datasets are also used in the evaluation study [MGAS09]. The data is enriched with uncertainty by replacing each object with a Gaussian *pdf* at the same position. Different variances are used in the experiments.

We measure the quality of the subspace clustering results via the F1 value, which is similar used in [MSE06, MGAS09]. We map each hidden cluster of a synthetic dataset to one identified cluster that matches the objects and the corresponding subspaces best. The harmonic mean of recall ('are all objects and dimensions detected?') and precision ('how accurate is the cluster detected?') is the F1 value of the cluster. The average of all clusters' F1 values is the quality value of the entire clustering. With this measure we account for the object groupings and the identified subspaces simultaneously. For real world data we use the class labels as indicators for the natural grouping of objects. No hidden subspaces are available in this case, and therefore the quality accounts only for the object groupings. The results of all algorithms are averaged over 5 runs. All experiments were conducted on computers with Opteron 2.3GHz CPUs and Java6 64 bit.

8.5.2 Evaluation of different variants.

In the following, we evaluate the different variants of our approach. That is, the support definitions and the differences between partitioning and non-partitioning clusterings based on these definitions.

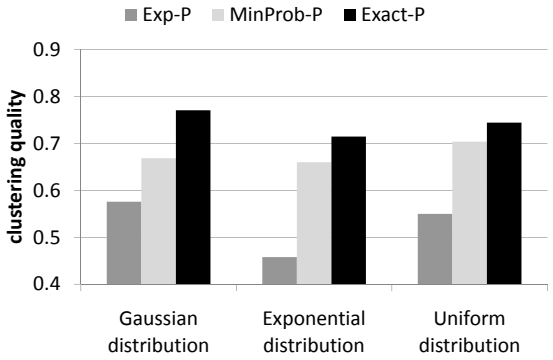


Figure 8.5: Comparison of support definitions (partitioning)

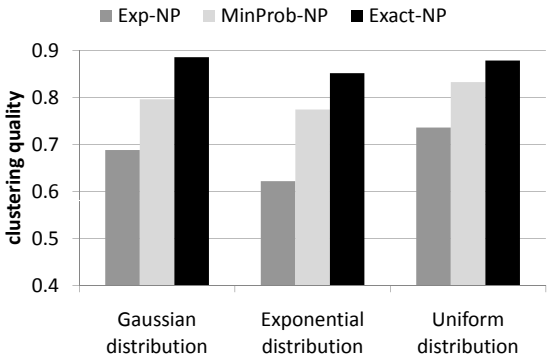


Figure 8.6: Comparison of support definitions (non-partitioning)

For partitioning clustering, Figure 8.5 compares the quality of the three support definitions for different underlying distributions of the uncertain objects, i.e. we used Gaussian, Exponential (skewed), or Uniform distributions for each object. As it was expected from our model, the qualities of the resulting clusterings improve with increasing usage of information. This is true for all three distributions. In comparison to the other two methods, Exp-P shows inferior performance and it is very sensitive to the chosen distribution; this is especially true for the skewed distribution. In Figure 8.6 the same experiment was repeated for the non-partitioning variants and confirms the conclusions drawn from Figure 8.5. As a result, we exclude Exp-P and Exp-NP in the rest of the experiments. A direct comparison of Figures 8.5 and 8.6 shows that overall the non-partitioning approaches generate clusterings of higher quality. This is also illustrated in Figure 8.7, where a varying overlap in the dataset, i.e. the percentage of objects that are assigned

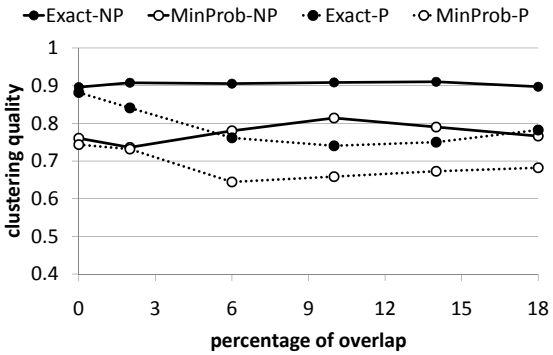
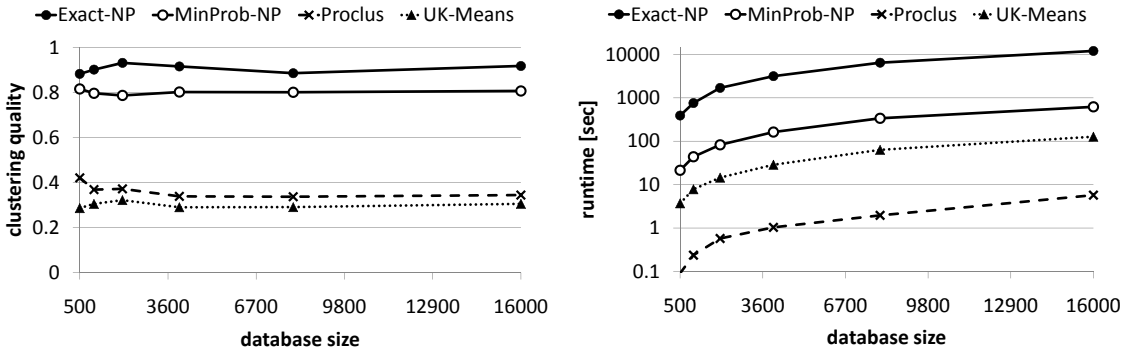


Figure 8.7: Varying overlap



(a) quality (b) efficiency

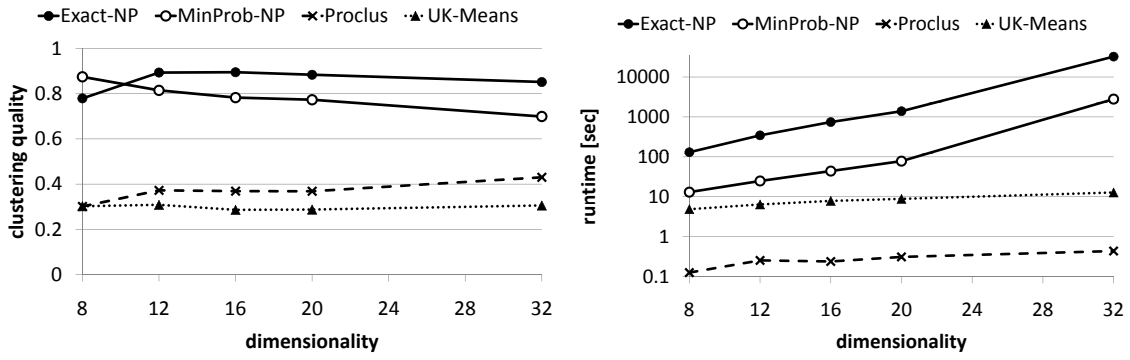
Figure 8.8: Varying size of the database

to more than one cluster, is analyzed. For no overlap (0%) the results of the partitioning (dotted lines) and non-partitioning (solid lines) variants are nearly the same. With increasing overlap both MinProb-NP and Exact-NP exceed their partitioning-based counterparts by more than 10%, and therefore we will only use the non-partitioning variants in the following experiments.

8.5.3 Comparison with competing approaches.

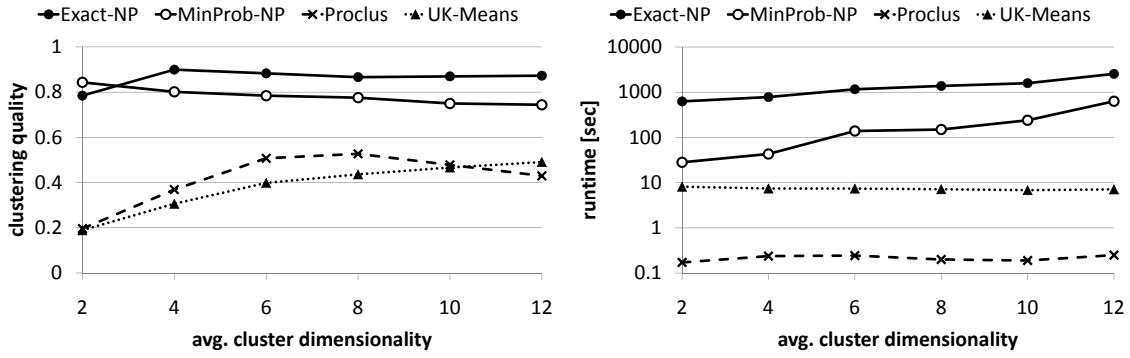
In the following we compare our methods to UK-Means and Proclus in several experiments on synthetic and real world datasets.

Database size is analyzed in Figure 8.8. The quality experiment shows that our approaches MinProb-NP and Exact-NP significantly outperform the competing algorithms by 45% and 55% respectively. For all algorithms the quality of clustering results are very stable, that is, the database size has no influence. Figure 8.8(b) compares the runtimes for the same experiment (Please note the logarithmic scale). The slopes of all 4 algorithms are similar, however, both UK-Means and Proclus are faster than our approaches, which is easily explainable: UK-Means only operates in the full dimensional space, i.e. no subspace projections are examined as in the other algorithms. Proclus uses certain values, but our approaches have to consider all the sample points of uncertain objects. Considering the bad quality of UK-Means and Proclus, their high efficiency is of no benefit. If we compare MinProb-NP to Exact-NP, an interesting observation can be made: Exact-NP takes more time but delivers better results.



(a) quality (b) efficiency

Figure 8.9: Varying dimensionality of the dataspace

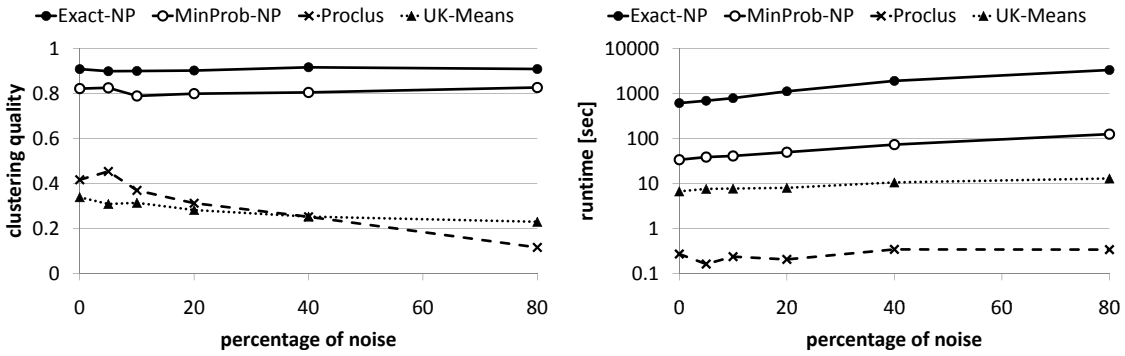


(a) quality (b) efficiency

Figure 8.10: Varying average dimensionality of clusters

Varying *dimensionality of the data space* is evaluated in Figure 8.9. The resulting clustering qualities are similar to the experiment in Figure 8.8(a), i.e. the competing algorithms are clearly outperformed by both of our methods by more than 50%. The runtimes also show a similar behavior as for database scalability. MinProb-NP and Exact-NP have higher qualities, and thus their runtime performances are again unproblematic.

The influence of a varying *average dimensionality of subspace clusters* in a 16-dimensional dataspace is examined in Figure 8.10. As before, the clustering qualities of Exact-NP and MinProb-NP exceed the results of the competing approaches significantly. While the results of our algorithms are very stable, the results of UK-Means and Proclus improve with a higher average dimensionality. Since UK-Means is a full-space clustering method, this is reasonable; with increasing dimensional-



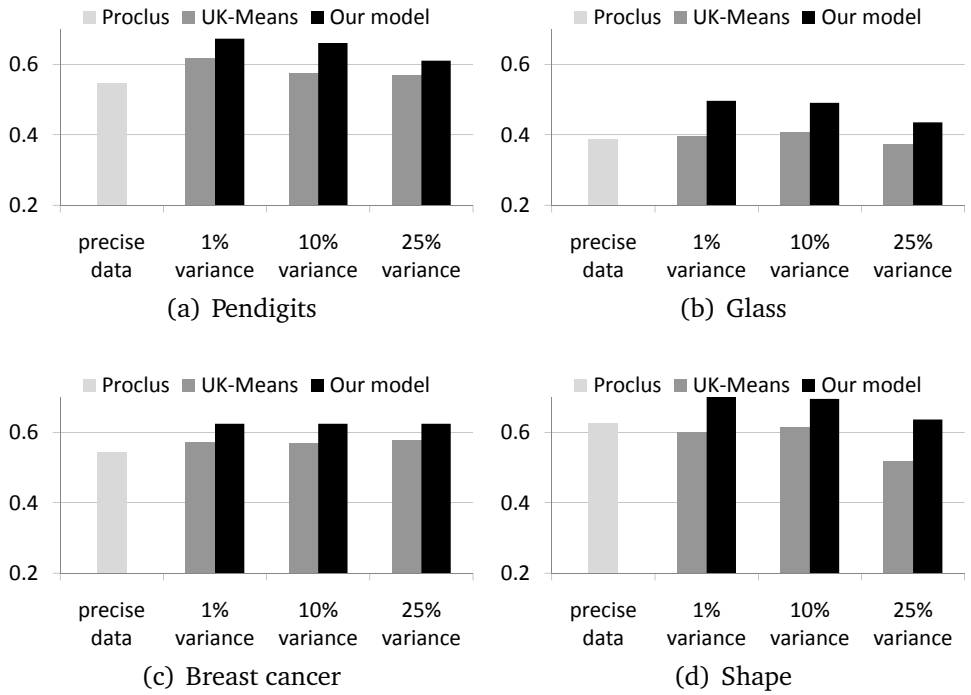
(a) quality (b) efficiency

Figure 8.11: Varying percentage of noise in the database

ity, more near-fullspace clusters are hidden in the data. For Proclus the results are rather surprising: because it is a subspace clustering algorithm, better results for lower average dimensionalities were expected. In contrast, our approaches are not influenced by the average dimensionality; the correct clusters are detected on all settings. The runtime evaluation in Figure 8.10(b) shows that the runtimes of the UK-Means and Proclus are relatively stable, while the runtimes of our algorithms increase with higher dimensionalities. This can be explained by the Apriori-based (cf. Section 8.4) procedure: for higher average dimensionalities, much more subspace candidates for each medoid have to be generated; for lower dimensionalities subspaces are pruned much earlier.

The influence of a varying *percentage of noise* in the dataspace, by increasing the number of objects that do not belong to any cluster, is analyzed in Figure 8.11. As in the other experiments, our methods show a much higher quality than UK-Means or Proclus. While our methods are very robust to noise, UK-Means and Proclus respond very sensitive: the quality decreases constantly with an increasing noise ratio of the dataspace. In Figure 8.11(b) the runtimes of all algorithms increase with more noise. The performance of our methods are more prone to it because with an increasing noise ratio, more medoids are randomly selected and discarded afterwards.

Performance of our model on *real world data* is analyzed in Figure 8.12. We present the results for the 4 datasets pendigits, glass, breast cancer, and shape. Note that we have optimized parameters for each algorithm on each dataset; for our model, we chose the best result from all methods. Proclus is executed on the



(a) Pendigits (b) Glass

(c) Breast cancer (d) Shape

Figure 8.12: Clustering quality on real world data

original precise data. Our model and UK-Means use the uncertain variants of the data; the variance of the underlying Gaussian distributions is set to 1%, 10%, and 25% of the data range.

The results on the pendigits dataset are presented in Figure 8.12(a). We can see that also on real world data our model outperforms the competing algorithms. Interestingly, the results of Proclus, operated on precise data, are worse than the results of the approaches that have to cope with uncertain information. For higher variances, however, we can see a decrease in quality; the clustering structure is obfuscated by the high variance and hence a detection of clusters is difficult.

On the remaining datasets similar results are obtained. Only the shape dataset (Figure 8.12(d)) is an exception: the quality of Proclus is slightly better than the quality of UK-Means. Nevertheless, for every dataset the effectiveness of our model is higher compared to the competing methods.

8.6 Conclusion

In this chapter we proposed a model that enables subspace clustering of uncertain data. We presented three variants that can be distinguished by the amount of information extracted from the probability density functions. Furthermore, we introduced a non-partitioning algorithm that is based on the concept of membership degree. We provided an efficient solution that uses Apriori-based pruning and heuristic sampling. Thorough experimental evaluation demonstrates that high-quality clusterings are generated and that competing algorithms are substantially outperformed.

This chapter concludes the second part of this thesis. We proposed integrated subspace clustering techniques that directly handle imperfect information and are robust with respect to errors in the data. Thus, we avoid the drawbacks accompanied by cleaning the data in a pre-processing step that is unaware of the characteristics of subspace clustering. While our novel models enable an effective subspace clustering for imperfect data, in the next part of this theses we tackle the challenges for mining heterogeneous data.

Part III

Subspace Mining on Heterogeneous Data

Chapter 9

Heterogeneous Data: Introduction

In the previous two parts of this thesis we have focused on subspace clustering techniques analyzing multi-dimensional vectors comprising numerical data. Even though this type of data is predominant, many applications can acquire multiple, diverse, and heterogeneous data sources. In this chapter we briefly discuss the various kinds of data available in today's applications and the resulting diversity of data mining methods, while the three subsequent chapters provide solutions for subspace clustering on a specific type of heterogeneous data.

9.1 Data Heterogeneity

The reasons for heterogeneous data are various: For example, heterogeneous data occurs by accessing a multitude of separate data sources, e.g. databases of different companies or departments. Each source might represent the objects with a different type of data leading to data heterogeneity. However, also in a single system, heterogeneous data might occur. Protein databases, for example, usually store for each protein its graph representation along with a textual description of its function. In other applications, heterogeneous data occurs because the original object representation, e.g. raw image data, cannot be processed reasonably but different feature extraction methods are required, e.g. the image is segmented or color features are extracted. Each feature might be of different type and, thus, leads to heterogeneous data.

In general, by analyzing heterogeneous data, we have to cope with a variety of different data types [HK01]. One of the most common data types is numerical vec-

tor data (either discrete or continuous) as discussed in the first part of this theses. Besides vector spaces, objects can often be described by categorical data, where one distinguishes between single-labeled data or set-labeled data. In the first case, each object is described by only a single value $l \in Dom$ of the label domain Dom. A typical scenario is the label domain of gender $Dom = \{female, male\}$. In the second case, each object is characterized by a subset $L \subseteq Dom$ of the labels, e.g. as given in the scenario of hobbies $Dom = \{tennis, soccer, baseball\}$.

More complex descriptions of objects include the representation by text data, e.g. web pages, by graph data, e.g. protein structures, or time series data, e.g. sensor measurements. Even though time series are often numerical, they differ from usual vector spaces since the time series' dimensions are subject to a specific temporal ordering.

In most applications, the previous data types are used to describe *individual* objects. The relations *between different* objects, however, are also a valuable information source. Such data can be observed in various domains including friendship relations in social networks, interacting genes in biology, or connected routers in the internet. Thus, besides characterizing single objects, network information is used to describe the objects' relations. Again, these relations can be represented by different types of data, such as binary relations or weighted relations.

Overall, data heterogeneity is not just a matter of the used data types, but also which information is described by the data: individual objects or relations between the objects. From the variety of possibilities, we analyze in this thesis binary network information in combination with numerical vector data since this is one frequently occurring type of heterogeneous information sources.

9.2 Mining Diversity

The diversity of heterogeneous data has lead to a diversity of research directions tackling specific challenges for the analysis of such kind of data.

Especially in the scenario where heterogeneous data is obtained by accessing a multitude of separate data sources, several challenges going beyond the ones of traditional data mining have to be solved: in this scenario it is not trivial to decide if two objects originating from different data sources describe the same or two different objects. If unique object identifiers, as the social security number, are available in all sources, a simple join step is sufficient. In general, however, more

complex data integration techniques such as schema mapping/matching [BV11, RB01] or entity resolution/record linkage [EIV07] are necessary. Furthermore, it is possible that objects do not occur in all sources, e.g. one source might represent the friendship relation between two persons, while in another source only one of these persons is listed.

Handling all these different scenarios is beyond the scope of this thesis. Thus, in the following we assume that the data integration is already performed, i.e. each object can uniquely be identified and it occurs in each source exactly once. Hence, we (implicitly) focus on the case where the data is recorded in a single system as, for example, given by a social network application where friendship relations are only possible between the unique users registered in the system.

Multi-Source Learning and Learning in Parallel Universes The majority of methods for an integrated mining of heterogeneous data focuses on the case where *individual* objects are represented in multiple different ways. In general, we can distinguish between the paradigms of *multi-source learning* [BS04, BM98, KKPS04] and *learning in parallel universes* [WHB10, CB10, WB07].

Let us note that the paradigm of multi-source learning is often called multi-view learning, which is motivated by the observation that for each object different representations, i.e. views, are stored. However, this paradigm is not to be confused with the multi-view methods introduced in Chapter 3. While the approaches discussed in Chapter 3 extract multiple clusterings from a single data source, the paradigm discussed in this section assumes multiple representations of the objects but extracts a single clustering (cf. next paragraph). Thus, to avoid confusion, we denote the paradigm discussed in this section as multi-source learning.

Multi-source clustering [BS04, BM98, KKPS04] tries to find a single global clustering solution that is of high quality in all given sources. For each of the different object representations, the groupings should be reasonable. Thus, multi-source clustering determines a consensus solution among all sources.

If many different sources are given, multi-source clustering might not be able to detect a global clustering result of high quality: some object representations might not provide valuable information for clustering, or some objects group better according to one representation, while another group is highly similar based on another representation. Thus, in contrast to multi-source clustering, clustering in parallel universes [WHB10, CB10, WB07], does not try to find groups that are

reasonable in all of the sources but the groups might be restricted to subsets of them.

The following example provides more insights in the above mentioned paradigms. Consider a database as given in the first part of this thesis, i.e. each object is represented by a d-dimensional vector. Instead of using this single source data, we now use each dimension as an individual source, i.e. we consider d many sources each representing the objects by 1-dimensional vectors. Multi-source clustering detects a clustering where each group must be reasonable in all sources. Thus, in the example it performs a kind of full-space clustering. In contrast, parallel universes is able to detect clusters which show up only in some of the sources. Thus, in the example it performs clustering in subspace projections.

In contrast to usual subspace/projected clustering, clustering in parallel universes is not restricted to vector data but each source might represent the object by a different type. Thus, it is designed to handle heterogeneous data. By detecting almost disjoint clusters, however, clustering in parallel universes is rather related to projected clustering than to subspace clustering. Accordingly, it shows the same limitations: the existing methods do not have the strength to detect multiple overlapping clusters or different views (cf. Chapter 3 and 4).

Our mining task Both previous paradigms in common is that they focus on heterogeneous data sources where objects are described individually. In this thesis, however, we analyze the more complex type where individual object descriptions as well as relations between different objects are given. We develop integrated clustering methods able to analyze vector data in combination with network data. This type of data is specifically challenging since the underlying mining models usually realize different and even contradicting objectives. We discuss the relevant related work for this specific mining task in detail in the next chapter.

In summary, in this thesis we analyze heterogeneous data which represents binary relations between the objects and describes individual objects by vector data. We develop integrated clustering methods that determine clustering solutions where each group is meaningful in the attribute domain as well as in the network domain. Thus, we exploit both sources. In the attribute domain itself, we perform subspace clustering. Thus, we do not need to consider all dimensions to detect the clusters.

Chapter 10

A Synthesis of Subspace Clustering and Dense Subgraph Mining

Today's applications deal with multiple types of information: graph data to represent the relations between objects and attribute data to characterize single objects. Analyzing both data sources simultaneously can increase the quality of mining methods. Recently, integrated clustering approaches were introduced, which detect densely connected node sets within one large graph that also show high similarity according to all of their attribute values. However, for attribute data it is known that this full-space clustering often leads to poor clustering results; one should prefer the task of subspace clustering.

In this chapter, we propose a method for finding homogeneous groups by joining the paradigms of subspace clustering and dense subgraph mining, i.e. we determine sets of nodes that show high similarity in subsets of their dimensions and that are as well densely connected within the given graph. Our *twofold clusters* should be maximized according to their density, size, and number of relevant dimensions. The optimization of the three objectives is usually conflicting; thus, we realize a trade-off between these characteristics to obtain meaningful patterns. To avoid redundant clusters in the result we develop a redundancy model to confine the clustering to a manageable size of only the most interesting clusters. Based on various pruning strategies we introduce the algorithm GAMER for the efficient calculation of our clustering. In thorough experiments on synthetic and real world data we show that GAMER achieves low runtimes and high clustering qualities.

10.1 Motivation

In the knowledge discovery process, clustering is an established technique for grouping similar objects while separating dissimilar ones. Traditional clustering methods group similar objects based on the information given by attribute data for each object. 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. Similarity of objects corresponds to similarity of attribute values. Another type of information is graph data, which represents relations between the objects. Based on this information, graph clustering methods determine groups of vertices that correspond to densely connected subgraphs within a single graph. For example, functional 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. Here, the similarity of objects corresponds to the density of connected subsets. Consequently, two different types of information can be used to determine clusters in the data.

In many applications information of both categories are available for the objects, graph information as well as attribute data. Such heterogeneous data can be modeled as a vertex-labeled graph such that vertices represent objects, edges represent relations between them, and feature vectors associated to the vertices represent the attributes for each object (cf. Fig. 10.1). Since the separate use of both data types can lead to highly differing clusterings and, thus, to contradictory results, the simultaneous use of both information for the process of clustering promises more meaningful and accurate results. Therefore, integrated clustering approaches handling this kind of heterogeneous data 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.

The main problem of almost all integrated 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. E.g. 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 [BGRS99]. Since clusters are strongly obfuscated by irrelevant dimensions and distances are not discriminable

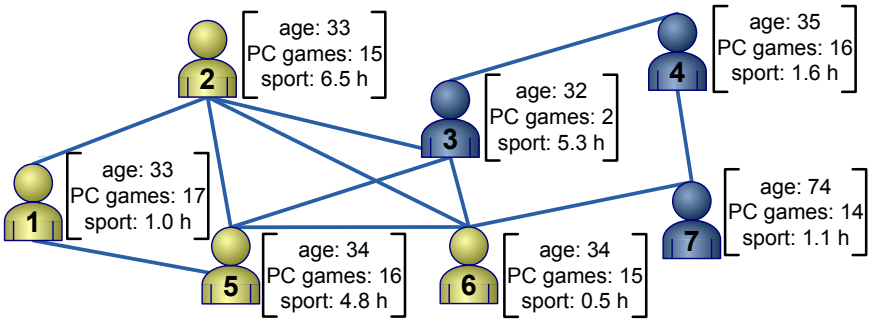


Figure 10.1: Combination of graph & attribute data and one potential twofold cluster (highlighted in yellow) with two relevant dimensions

any more, searches in the full-space are futile or lead to very questionable clustering results. As discussed in this thesis, subspace clustering provides a solution to this problem. Consistently, also integrated clustering models should analyze subsets of the attributes. However, this aspect is not adequately considered by the existing models.

Our novel approach combines graph data and attribute data to identify groups according to their density of connections as well as values of dimensions. In contrast to other approaches, however, we consider subsets of the dimensions to realize meaningful similarity determination. In Fig. 10.1 for example we are able to identify the cluster $\{1, 2, 5, 6\}$ because the objects are similar in 2 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, to 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 integrated clustering models focus on graph properties as determining maximal sets whose density is large enough. In Fig. 10.1 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 and 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 equal consideration. This enables each paradigm to be on a par with the other one 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 2 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 and, thus, a strong overlap usually indicates redundancy. As shown in the previous chapters of this thesis, considering redundancy is indispensable for subspace clustering methods. Also in the field of graph mining, avoiding redundant patterns is studied [HCS+07]. The importance of a proper treatment of redundancy is hence increased for the combined consideration of subspace clustering and subgraph mining albeit rarely treated accurately in the past. Our model successfully avoids redundancy in the clustering result, while generally allowing the clusters to overlap.

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 limitations in the analysis of attribute and graph data.

10.2 Comparison with Related Work

In the past decades different clustering and subgraph mining paradigms have been proposed, which we will shortly review in this section.

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 [BGRS99, HK01]. Consequently, in this thesis we consider *subspace clustering methods* detecting clusters hidden in individual subspace projections of the data [KKZ09, PHL04]. This is also important for the analysis of heterogeneous network data, because we cannot expect that densely connected groups are similar in all attributes but 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. As shown in the previous chapters of this thesis, 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 use *redundancy models* to confine the output to the most interesting clusters and thus get a reasonable result size (cf. especially Chapter 2 and 3). None of the proposed subspace clustering methods, however, considers graph data.

Clustering graph data. Clustering *graph data* has been done in different ways [AW10]: For our purpose of finding groups in networks we focus on methods mining densely connected subgraphs in one large graph. Methods as [LWZY06, LZY07, RZ07, DHZ$^+$01] perform a graph-partitioning while others assume that the given graph naturally divides into (possibly overlapping) subgraphs of certain patterns, e.g. cliques [WZZ06] or γ-quasi-cliques [LW08, PJZ05, ZWZK06, ARS02]. 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

[HCS+07, ZYL09]. However, in the field of graph clustering/dense subgraph mining the problem of redundancy has not yet been researched thoroughly.

Integrated clustering approaches. Apart from the previously seen approaches there exist methods that consider both information types: vector data and graph data. Some of these techniques [DWP+07, KMS03, ITM06] consider attribute data only in a post-processing step after determining subgraphs. Thus, attributes do not influence the resulting subgraph structures.

In [HZZL02] 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 afterwards. Using combined distance functions leads to results that are difficult to interpret as no conclusions about cluster structures in the graph are possible. Similarly, the methods of [STM07, NAJ04] 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 [HZZL02], which transforms the graph to distance values, the approach by [US07] 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 [EGG+06] extends the k-center problem by requiring that each group has to be a connected subgraph. All the previous approaches [HZZL02, STM07, NAJ04, US07, EGG+06] 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 [BGRS99], they unconsciously use only the graph information for clustering, making the desired combination meaningless. Furthermore these approaches determine disjoint, or almost disjoint, clusters.

In the work of [ZCY09, ZCY10] 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 correspond in all attributes, they are only pairwise similar and no specific relevant attribute subset can be defined for the clusters. In [SJZ10] 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 [MCRE09] 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 integrated clustering methods, one major drawback is their missing or limited redundancy handling. Especially for approaches analyzing subsets of attributes, as CoPaM for the integrated 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 novel approach, redundant clusters are removed from the output.

10.3 An Integrated 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 methods for 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 to the set of vertices $O \subseteq V$ and $x[i]$ to refer to the i-th component of a vector $x \in \mathbb{R}^d$.

Based on this input graph, we introduce in Section 10.3.1 the cluster definition that defines the properties a single valid cluster has to fulfill. In Section 10.3.2 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.

10.3.1 Cluster definition

Our twofold clusters should represent meaningful subspace clusters and at the same time meaningful dense subgraphs. Therefore we combine established defini-

tions of both paradigms. For subspace clustering the cell-based methods show high quality results and they are efficiently computable [MGAS09]. Thus, we choose this paradigm for our model and use a cluster definition similar to the one used by [PJAM02]. 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 10.1 *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 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. 10.2 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 [LW08]. 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.

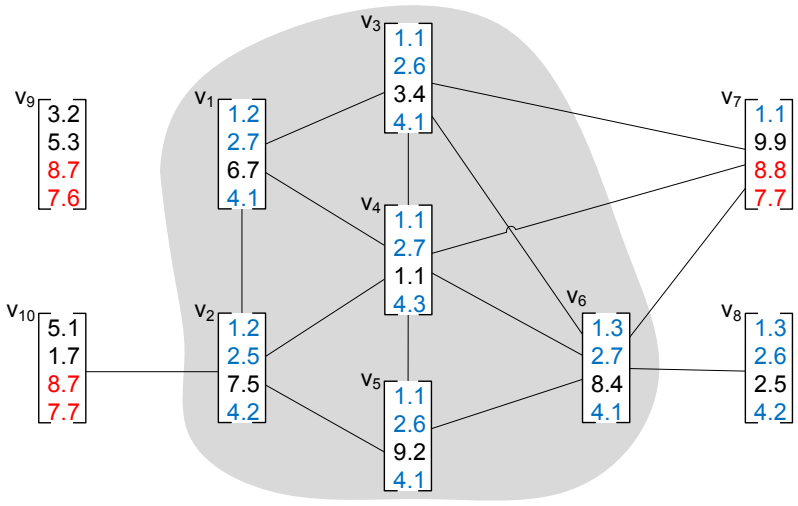
Definition 10.2 *Quasi-clique property*
A set of vertices $O \subseteq V$ within a graph $G = (V, E, l)$ is a γ-quasi-clique if

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

where $deg^O(v)$ 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. 10.2 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.



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

Figure 10.2: Exemplary groups and their properties

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 10.3 *Twofold cluster*
A twofold cluster $C = (O, S)$ within a graph $G = (V, E, l)$ 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 [ZWZK06]. For a twofold cluster, however, the connectivity is reasonable and enforced by our definition. For example in Fig. 10.2 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 10.3 we get more meaningful clusters.

10.3.2 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. 10.1). E.g., by choosing $s_{min} = 1$ the cluster (O_2, S_2) in Fig. 10.2 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 Section 10.3.2 and our redundancy model in Section 10.3.2. In Section 10.3.2 the overall clustering is defined.

Quality function To confine the $Result \subseteq Clusters$ to the most interesting clusters, we need a measure for the quality. Measuring the interestingness of our twofold clusters is not trivial: 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 10.4 *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 the example in Fig. 10.2 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 $Q(C_3) = 10.8$.

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 (w.r.t. 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 C's quality is lower. If the cluster C had a higher quality, then it should not be reported as redundant w.r.t. C'; the user is more interested in C. Thus, $Q(C) < Q(C')$ must hold for the redundancy of C w.r.t. C'.

Furthermore, the cluster C induces redundancy w.r.t. C' if it does not describe novel structural information. In our context, the objects as well as the relevant dimensions of $C = (O, S)$ have already been covered to most parts 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 conditions hold, the cluster C is redundant w.r.t. C'. We denote this by $C \prec_{red} C'$ and we formally define:

Definition 10.5 (Binary redundancy relation)
Given the redundancy parameters $r_{obj}, r_{dim} \in [0, 1]$, the binary redundancy relation $\prec_{red}$ is defined by:

$$\text{For all twofold clusters } C = (O, S), C' = (O', S'):$$
$$C \prec_{red} C' \Leftrightarrow \left[Q(C) < Q(C') \quad \wedge \quad \frac{|O \cap O'|}{|O|} \geq r_{obj} \quad \wedge \quad \frac{|S \cap S'|}{|S|} \geq r_{dim}\right]$$

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' to denote C as redundant. 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. 10.2 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 conditions have to hold. It is not enough that, e.g., only the set of objects is covered; if considerably many other dimensions are comprised, the cluster will still not be redundant. Especially, two clusters located in disjoint subspaces are never redundant to each other (if $r_{dim} > 0$) leading to the possibility to detect multiple clustering views.

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, the result would just be $\{C_c\}$. However, the result $\{C_a, C_c\}$ is more useful since these clusters are also pairwise non-redundant and we additionally get C_a. Intuitively, since we exclude C_b from the result, the property $C_a \prec_{red} C_b$ does not need to be considered; thus, C_a represents novel information and it should be included in the result.

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 10.6 *Optimal twofold clustering*
Given the set of all twofold clusters $Clusters$, the optimal twofold clustering $Result \subseteq Clusters$ fulfills

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

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.

10.3.3 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 $SELECT$.

Complexity of the overall result determination

Theorem 10.1 *Given a vertex-labeled graph $G = (V, E, l)$, determining the optimal clustering according to Def. 10.6 is #P-hard w.r.t. 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 $MAXCLIQUE$ [GJ79] 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 10.1
Input Mapping: *The graph G is taken as it. We 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.*

$$C = (O, S) \prec_{red} C' = (O', S')$$

$$\Leftrightarrow Q(C) < Q(C') \wedge \frac{|O \cap O'|}{|O|} \geq 1 \wedge \frac{|S \cap S'|}{|S|} \geq 0$$

$$\Leftrightarrow |O| < |O'| \wedge O \subseteq O' \Leftrightarrow O \subset O'$$

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 $MAXCLIQUE$.* □

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 10.2 *Given a set of twofold clusters $Clusters$, the optimal clustering according to Def. 10.6 can be determined in quadratic time w.r.t. the number of clusters.*

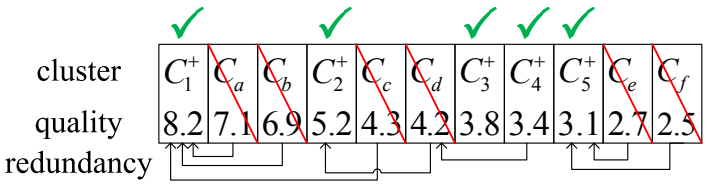


Figure 10.3: Ranking of clusters to determine the optimal clustering

To prove this theorem we provide an algorithm calculating the optimal result. Let us 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^+. Because 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 that are redundant to C_2^+. These steps can be repeated until no clusters remain.

Thus, our optimal twofold clustering can be calculated with the procedure that is illustrated in Figure 10.3. At the beginning, our result set is empty ($Result = \emptyset$) and clusters are ranked in descening 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 10.1 *The clustering result based on Definition 10.6 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 $\mathcal{O}\left(|Clusters| \cdot log\left(|Clusters|\right)\right)$. In the second phase, each cluster is compared to the current result set to evaluate the redundancy. In the worst case, each clus-

ter 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:

Theorem 10.3 *Given a set of twofold clusters $Clusters$, the optimal clustering result $Result \subseteq Clusters$ according to Def. 10.6 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 w.r.t. the number of clusters, however, #P-hard w.r.t. the number of vertices, the number of clusters has to depend superpolynomial 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 10.2 *Given a vertex-labeled graph $G = (V, E, l)$, the number of twofold clusters $Clusters$ cannot be polynomially bounded w.r.t. the number of vertices V.*

10.3.4 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. 10.4). 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. 10.5). 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 w.r.t. their objects or w.r.t. 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. Similar to our OSCLU and MVGen model (cf. Chapter 3 & 4), we are able to detect multi-views: if the subspaces are disjoint, completely different groupings are possible.

Last, the user has to specify the characteristics each individual cluster has to fulfill (cf. Def. 10.3). 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, are anyhow not selected due to our redundancy model. Thus, if no further knowledge is given one could simply ignore the thresholds (at the cost of increased runtime since more clusters need to be analyzed). Since often, however, some basic properties of the clusters are desired, e.g. clusters with two objects or clusters located in 1-dimensional subspaces are usually not meaningful, one should restrict the set of valid clusters a-priori by the three thresholds and far lower runtimes of the overall algorithm are achievable. 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 [YM05, PJAM02].

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.

10.4 The GAMER Algorithm

In the following section we introduce our algorithm GAMER (**G**raph & **A**ttribute **Mi**n**er**) 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 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.

10.4.1 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 10.3 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.

Initial pruning

Based on our subspace cluster property, two neighboring vertices can only belong to the same cluster if they are similar (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. vertices with

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

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.

Efficient enumeration of twofold clusters

To enumerate the vertex sets in our graph, we use the set enumeration tree [Rym92]. A complete tree for a graph with four vertices is depicted in Fig. 10.4. Each node of the tree represents a set 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. 10.4 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 Section 10.4.3.

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 [PJZ05], 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. 10.4 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.

Pruning by subspace diameter

In [PJZ05] 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

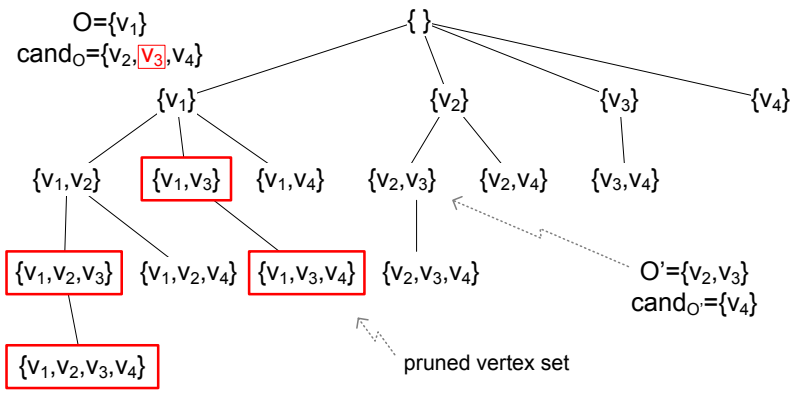


Figure 10.4: Exemplary set enumeration tree and pruned vertex sets

by

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

where $d_V(t,v)$ 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 loose 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 by [LW08]). 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.

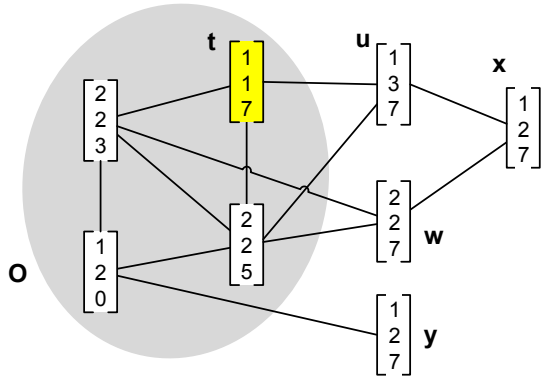


Figure 10.5: Pruning of vertices based on diameter and neighborhood calculations

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. 10.5. 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 w.r.t. 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 the 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. However, as the vertex u is excluded from the candidate set this path does not longer exist in the new candidate set; thus the vertex x 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 10.7 *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.

Pruning by subspace vertex degrees

Our algorithm uses several pruning techniques based on the vertex degrees. In traditional quasi-clique mining [LW08] 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 w.r.t. the sets O and $cand_O$. In traditional quasi-clique mining these two definitions are given as

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

Considering Fig. 10.6 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), the exdegree describes the number of adjacent vertices in the candidate set $cand_O$

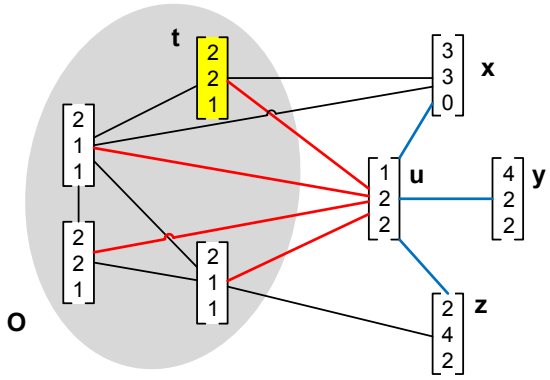


Figure 10.6: Example for novel indegree/exdegree definitions based on subspaces

(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 [LW08, ZWZK07] 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 w.r.t. subspaces The basic idea of our pruning is introduced by the example in Figure 10.6. Considering the traditional – just graph based – definitions for the indegree and exdegree we get in Figure 10.6 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 w.r.t. 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 selected the second dimension, e.g. the subspace $\{1, 2\}$, the subspace property would be violated by $O \cup \{x\}$. Thus, intuitively the indegree for the vertex x is zero or undefined for subspaces including the dimension 2.

Considering the traditional exdegree: the vertex 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 the 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, the vertex 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, theoretically we can determine for each subspace $S \subseteq S(O)$ with $|S| \geq s_{min}$ an individual indegree and exdegree.

Definition 10.8 *Subspace indegree and exdegree*
The subspace indegree and the subspace exdegree of a vertex u w.r.t. the vertex set O, the candidate set $cand_O$, and the subspace S are

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

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

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

Note that S is not necessarily the current subspace $S(O)$ of O but can be any subspace $S \subseteq S(O)$ with $|S| \geq s_{min}$ since the true subspaces of the potential clusters are not known at this time. 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. 10.6 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 subspace S, 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 have to be similar in the current subspace since otherwise we cannot select both vertices simultaneously for the 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 [LW08, ZWZK07] we need a single indegree/exdegree value per object. Our novel definitions determine the degree values if the subspaces of the potential clusters are known. However, since we do not know the subspaces of the clusters $C = (X, S)$ with $O \cup \{u\} \subseteq X \subseteq O \cup cand_O$ apriori[1] and since we have to guarantee that vertices belonging to valid clusters are not pruned, we have to perform a optimistic estimation for the overall degrees of a vertex u: We have to choose the subspace S that leads to the *highest* indegree and exdegree values respectively, because the pruning techniques exclude vertices if their $indeg$ and $exdeg$ values are too low. Thus, within the pruning method introduced beforehand we could use the values

$$indeg^O_{new}(u) = \max_{S \subseteq S(O), |S| \geq s_{min}} \{indeg^O(u, S)\}$$

$$exdeg^O_{new}(u) = \max_{S \subseteq S(O), |S| \geq s_{min}} \{exdeg^O(u, S)\}$$

without losing any valid vertex.

Obviously, determining these values is exponential w.r.t. the size of the current subspace $S(O)$ and thus computationally demanding. To still guarantee an

[1] We just know that $S \subseteq S(O)$ and $|S| \geq s_{min}$ has to hold.

efficient pruning method, we define upper bounds for the degree values that are computable in linear time.

We compute one indegree and exdegree value per dimension $d \in S(O)$ which provides an upper bound for the corresponding value for any subspace containing the dimension d. As the subspace of a valid cluster has to contain at least s_{min} dimensions we choose the s_{min}-highest value as the indegree or exdegree of the vertex:

Definition 10.9 *Linear time subspace indegree and exdegree*
The linear time subspace indegree and subspace exdegree of a vertex u w.r.t. the vertex set O and the candidate set $cand_O$ are

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

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

For vertex u in our example we would get $indeg^O_{lin}(u) = 4$ 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 $exdeg^O_{lin}(u) = 1$. 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. We prove the following inequalities:

Theorem 10.4 *Bounds for the indegree and exdegree values*
Given a set O and the corresponding candidate set $cand_O$, the following bounds hold for each $u \in O \cup cand_O$:

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

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

Proof 10.2 *1) We first show $indeg^O_{new}(u) \leq indeg^O_{lin}(u)$: Obviously, $indeg^O(u, S) \leq indeg^O(u, \{d\})$ for all $d \in S$ since less constraints are given for the attribute values of u. Thus, also $indeg^O(u, S) \leq \min_{d \in S} indeg^O(u, \{d\})$ holds and we infer:*

$$indeg^O_{new}(u) = \max_{S\subseteq S(O),|S|\geq s_{min}} \{indeg^O(u,S)\} \leq$$

$$\max_{S\subseteq S(O),|S|\geq s_{min}} \left(\min_{d\in S} indeg^O(u,\{d\})\right) = \max_{S\subseteq S(O),|S|=s_{min}} \left(\min_{d\in S} indeg^O(u,\{d\})\right) =$$

$$\max_{S\subseteq S(O),|S|=s_{min}} \left(indeg^O(u,\{d^*\}) \textit{ with } d^* = \operatorname*{argmin}_{d\in S} indeg^O(u,\{d\})\right) =$$

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

2) Next, we show the inequality $indeg^O_{lin}(u) \leq indeg^O(u)$*: From the definition of* $indeg^O_{lin}(u)$ *we can derive*

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

$$\leq \max_{d\in S(O)} indeg^O(u,\{d\}) \overset{\textit{Def. 10.8}}{\leq} \max_{d\in S(O)} |\{x\in O \mid (u,x)\in E\}| = indeg^O(u)$$

3) For the exdegree, the inequality $exdeg^O_{new}(u) \leq exdeg^O_{lin}(u)$ *is shown by*

$$exdeg^O(u,S) \leq \min_{d\in S} exdeg^O(u,\{d\})$$

which can be derived from Def. 10.8. It holds $exdeg^O(u,S)$

$$\leq |\{x\in cand_O \mid (u,x)\in E \wedge indeg^O(x,S)\neq -1 \wedge \forall d\in S : |u[d]-x[d]|\leq w\}|$$

$$\leq |\bigcap_{d\in S}\{x\in cand_O \mid (u,x)\in E \wedge indeg^O(x,\{d\})\neq -1 \wedge |u[d]-x[d]|\leq w\}|$$

$$\leq \min_{d\in S} exdeg^O(u,\{d\})$$

The construction of the subspace S *which maximizes* $exdeg^O(u,S)$ *can be done as in the proof for the indeg value.*

4) Last, we prove $exdeg^O_{lin}(u) \leq exdeg^O(u)$*:*

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

$$\leq \max_{d\in S(O)} exdeg^O(u,\{d\}) \overset{\textit{Def. 10.8}}{\leq}$$

$$\max_{d\in S(O)} |\{x\in cand_O \mid (u,x)\in E \wedge indeg^O(x,S)\neq -1 \wedge \forall d\in S : |u[d]-x[d]|\leq w\}|$$

$$\leq \max_{d\in S(O)} |\{x\in O \mid (u,x)\in E\}| = exdeg^O(u)$$

□

The inequalities $indeg^O_{new}(u) \leq indeg^O_{lin}(u)$ and $exdeg^O_{new}(u) \leq exdeg^O_{lin}(u)$ have to hold since otherwise we potentially prune vertices which still could lead to valid clusters. The other two 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.

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.

10.4.2 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.

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 Section 10.3.3, we can use a ranking of clusters to enable the efficient determination of

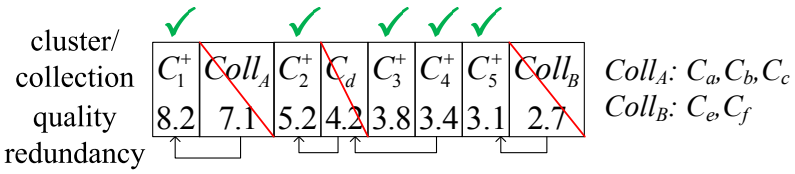


Figure 10.7: Extended ranking that considers clusters and cluster collections

our optimal clustering (cf. Figure 10.3). 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 whole 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 initially those cluster collections $Coll_i$ from validity considerations for which a cluster C_r exists such that *all* clusters represented by one $Coll_i$ are redundant 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. We do not have to generate and verify any of the represented clusters. This idea is illustrated in Fig. 10.7. 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 10.10 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 10.10 *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* $cand_O$
- *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* $C \Rightarrow C \prec_{red} C_r$

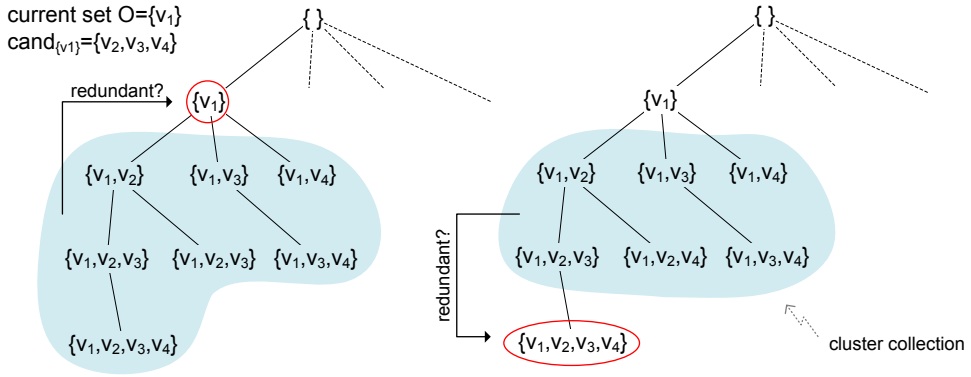


Figure 10.8: Two types of cluster collections: superset collection (left) and subset collection (right)

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 w.r.t. 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. 10.8. 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 w.r.t. 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 w.r.t. 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 10.5 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.

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 10.5 *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* $min_deg = \min_{v \in O}(indeg_{lin}^O(v) + exdeg_{lin}^O(v))$

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

Proof 10.3 *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:

$$\min_{v \in X}(deg^X(v)) \leq \min_{v \in O}(indeg^O_{lin}(v) + exdeg^O_{lin}(v)) = min_deg$$

We only consider vertices from O as the vertices from $cand_O$ do not have to be part of X, 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

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

□

Subset collection The second pruning technique is illustrated on the right hand side of Fig. 10.8. 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 w.r.t. 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 w.r.t. 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 10.5 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 10.6 *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* $min_deg = \min_{v \in O}(indeg_{lin}^O(v) + exdeg_{lin}^O(v))$

c) $|X| \leq |O \cup cand_O| - 1 =: n_{max}$

Proof 10.4 *a) As in the proof of Theorem 10.5 a)*

b) As in the proof of Theorem 10.5 b)

c) For every set $X \subset (O \cup cand_O)$ it holds that $|X| \leq |O \cup cand_O| - 1$.

□

Pruning of cluster collections w.r.t. non-anchor clusters

The previous techniques avoid the processing of a collection if all possible clusters in this subtree would be redundant w.r.t. 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 w.r.t. 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 are searching in the current result for a cluster $C^* = (O^*, S^*)$ so that each cluster from the collection is redundant to C^*. For this, the three properties according to Definition 10.5 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 w.r.t. 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 w.r.t. the objects, any cluster of the collection has to be redundant w.r.t. the objects. We show:

   Theorem 10.7 *If*

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

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

   Proof 10.5 *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 much vertices from* $cand_O \setminus O^*$ *as possible. As* $X \supset O$*, it also holds that* $(X \cap O^*) \supset (O \cap O^*)$*, 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* $cand_O \setminus O^*$ *to* X*. However, if* $|O| + |cand_O \setminus O^*| < n_{min}$ *we have to add*

$n_{min} - |O| - |cand_O \setminus O^*|$ 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

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

□

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 w.r.t. the dimensions, any cluster of the collection has to be redundant w.r.t. the dimensions. We show:

   Theorem 10.8 *If*

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

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

   Proof 10.6 *For the subspace S of any cluster from the collection it holds that $S \subseteq S(O)$ (antimonotonicity). 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 $S(O) \setminus S^*$ as possible. If $|S(O) \setminus S^*| < s_{min}$ we have to add $s_{min} - |S(O) \setminus S^*|$ 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 $s_{min} - |S(O) \setminus S^*|$. Thus the minimal value for $\frac{|S \cap S^*|}{|S|}$ is $\frac{s_{min} - |S(O) \setminus S^*|}{s_{min}}$.* □

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 w.r.t. C^*.

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.

10.4.3 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 Algorithm 10.1 and Algorithm 10.2. 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. 10.2) clusters and/or cluster collections are inserted into the queue. At the beginning, we prune the candidate set of the current node (line 1) according to the techniques introduced in Section 10.4.1. 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 4), which is sorted according to the corresponding quality values. After this, we apply our pruning based on cluster collections: We generate the collections (line 5 and 14) and we check their redundancy w.r.t. the clusters C_O or C_r respectively (line 8 and 15). 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 descent into the subtrees and recursively invoke the traversal function for each vertex in the pruned candidate set (line 20). Please note that the candidate set is sorted based on the indegree ($indeg^O_{lin}(u)$) 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. 10.1). 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 w.r.t. 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

Algorithm 10.1: 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 10.4.1
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. 10.4.2)
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_lin(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
```

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

Algorithm 10.2: GAMER algorithm ($DFS_traversal$ method)

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

```
determine and prune cand_O (based on cand_O') // cf. Section 10.4.1
determine C_O = (O, S(O)) // current cluster
if C_O is valid twofold cluster then
    insert C_O into queue
    determine superset collection Coll_sup
    if n_max < n_min or γ_max < γ_min then
        return; // no further clusters in this subtree possible
    if Coll_sup is redundant to C_O then
        insert Coll_sup into queue
        return; // (currently) stop DFS traversal
determine C_r = (O ∪ cand_O, S(O ∪ cand_O)) // look ahead
if C_r is valid twofold cluster then
    insert C_r into queue
    determine subset collection Coll_sub
    if Coll_sub is redundant to C_r then
        insert Coll_sub into queue
        return; // (currently) stop DFS traversal
sort vertices v ∈ cand_O descendingly by indeg^O_lin(v)
for v ∈ cand_O do
    DFS_traversal(O ∪ {v}, cand_O)
    cand_O = cand_O \ {v} // consider only vertices ordered behind v
```

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. Based on our model-specific characteristics we realize an efficient execution.

10.5 Experimental Analysis

10.5.1 Experimental setup

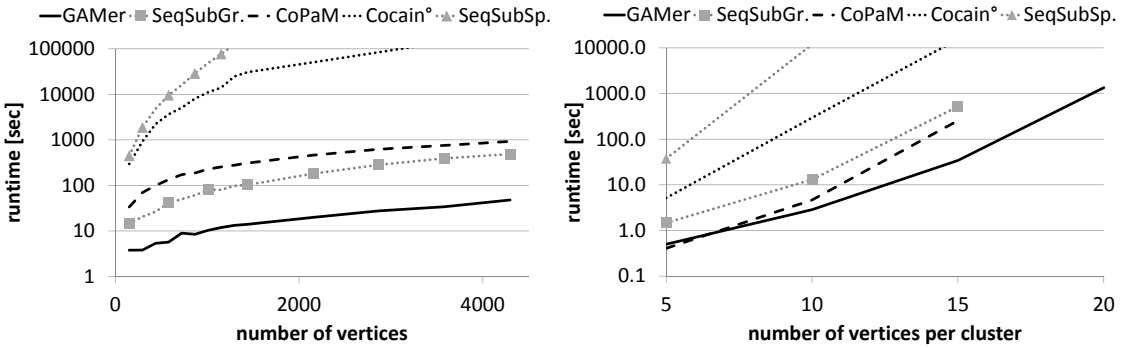
We compare our GAMER method to CoPaM [MCRE09], the only method which also considers subspaces and dense subgraphs. To include a further competitor we extend Cocain [ZWZK06] to handle our data. Originally, this method considers a set of graphs to find sets of vertices forming quasi-cliques in several 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 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 [LW08]) 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 afterwards 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 $r_{obj} = r_{dim} = 0.5$.

We use synthetic data and several publicly available real world data sets. 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 [S$^+$06, S$^+$05] (3548 vertices; 8334 edges; 115 dimensions), *patent information*[2] (492007 vertices; 528333 edges; 5 dimensions), an extract of the *Arxiv database*[3] (13003 vertices; 120213 edges; 300 dimensions) and a co-author graph extracted from the *DBLP database*[4] (2482 vertices; 7438 edges; 11 dimensions). The synthetic data is generated based on quasi-cliques since all competing approaches perform quasi-clique detection. To model such data, we first generate quasi-cliques of certain sizes and densities that represent the ground truth information. Then, these vertex sets are randomly connected by further edges and feature vectors are assigned based on the methods of [PJAM02, YM03]. We additionally implement that the vertices of different quasi-cliques can overlap. By default we generate 20 dimensional data with 80 clusters, each with 15 vertices, an average density of 0.6 and 5-10 relevant dimensions. 6% of the clusters vertices overlap. We vary the different characteristics of the data in our experiments.

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

[2]http://www.nber.org/patents/
[3]http://www.cs.cornell.edu/projects/kddcup/datasets.html
[4]http://dblp.uni-trier.de



(a) Varying number of clusters (b) Varying number of vertices per cluster

Figure 10.9: Scalability of the algorithms with respect to the database size

evaluation of subspace clustering [MGAS09]. 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.

10.5.2 Scalability on synthetic data

In the first experiment in Fig. 10.9(a) 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 data sets 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. 10.9(b) 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 data sets. The various pruning strategies of GAMER enable the efficient generation of the clustering solution.

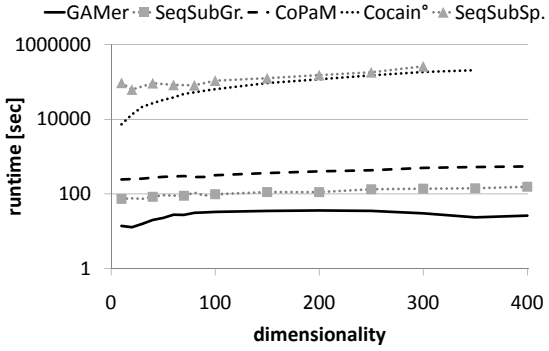


Figure 10.10: Scalability of the algorithms with respect to the database dimensionality

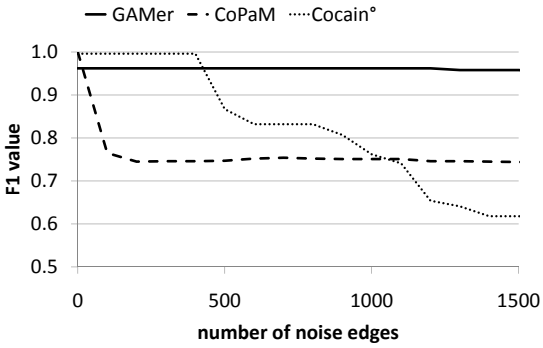


Figure 10.11: Clustering quality vs. noise

Fig. 10.10 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 better because the absolute runtime of GAMER is still the lowest, as also indicated by the previous experiments.

10.5.3 Quality on synthetic data

In the following experiments, we exclude the baseline algorithms because the clustering results are identical to GAMER. In Fig. 10.11 we analyze the robustness of the algorithms w.r.t. noise. Besides the clustered objects we add noise objects that do not belong to any cluster (25% w.r.t. 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$^\circ$ 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 w.r.t. 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 few noise; however, by trading off the three characteristics ($a{=}b{=}c{=}1$) we get high qualities for all settings.

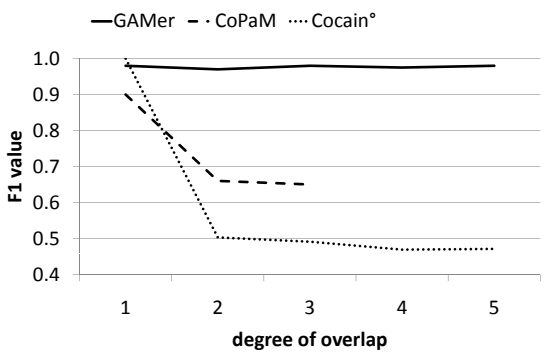


Figure 10.12: Clustering quality vs. overlap

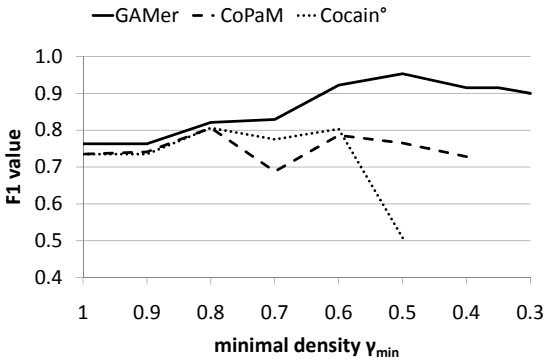


Figure 10.13: Clustering quality vs. varying minimal density γ_{min}

In Fig. 10.12 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 are obtained. Our redundancy model, however, prevents to generate these clusters.

Next, we analyze the effects of a varying minimal density γ_{min}. In Fig. 10.13 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 also 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 w.r.t. the minimal density. An-

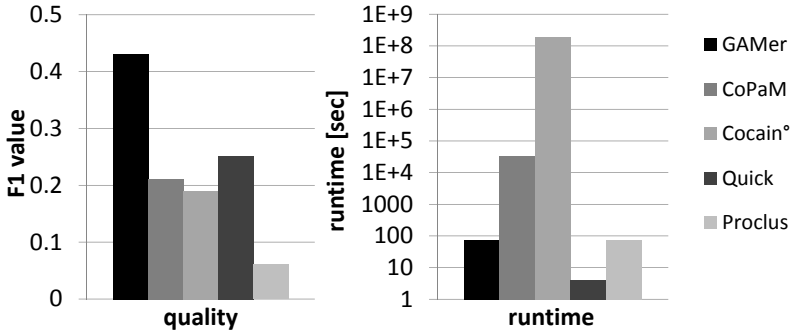


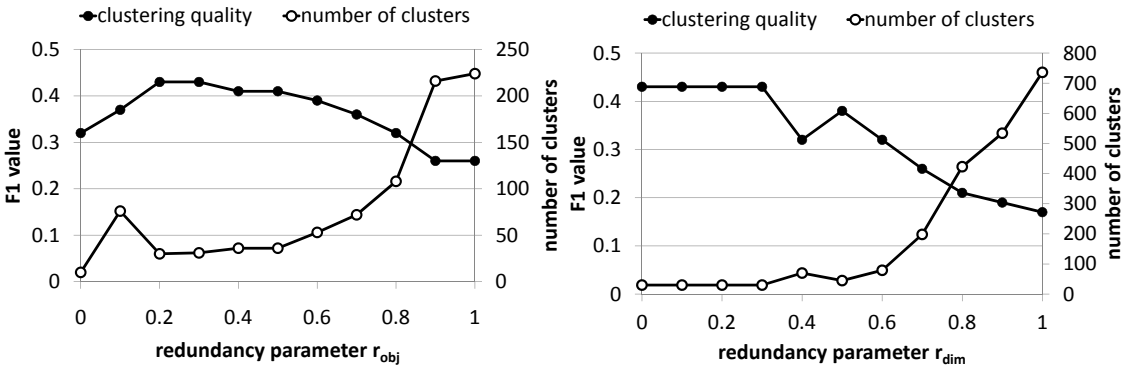
Figure 10.14: Clustering quality and runtime of the algorithms for gene data

other 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 gets a high clustering quality by confining the result to the most interesting and non-redundant clusters.

10.5.4 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 [ZFW+03] that assigns genes to biological categories. These classes are used as hidden clusters as also done in [MCRE09]. For this experiment in Fig. 10.14, 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 [LW08]) or subspace clustering (Proclus [AWY+99]). The quality of these two algorithms is low, indicating that a synthesis of both paradigms – as 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 data set. In Fig. 10.15(a) 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



(a) Influence of parameter r_{obj}

(b) Influence of parameter r_{dim}

Figure 10.15: Influence of the redundancy model on the clustering (gene data)

many and also important clusters. In Fig. 10.15(b) 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 data sets (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. 10.16 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 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. 10.16 indicates that our approach generates clusters with just slightly less objects than Co-

	Gene		Arxiv		DBLP			Patent
	GAMER	CoPam	GAMER	CoPam	GAMER	CoPam	Cocain°	GAMER
runtime [s]	76	33060	22	4440	6	22589	3963	574
# clusters	30	115581	23	98419	83	341333	151	146
∅ vertices	8.8	9.67	9.04	9.12	10.1	10.8	9.6	11.7
∅ density	0.72	0.24	0.64	0.23	0.62	0.2	0.62	0.6
∅ dimensionality	15.47	12.21	5.04	5.0	3.0	3.0	3.0	3.0

Figure 10.16: Clustering properties for real world data (not listed methods did not finish within 2 days)

PaM that concentrates on maximizing the number of vertices per cluster. Although GAMER implements a trade-off between different criteria it determines clusters of comparable size but considerably higher density. Especially for the gene data, we see that the dimensionality is higher too.

In Fig. 10.17 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. Fig. 10.17 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.

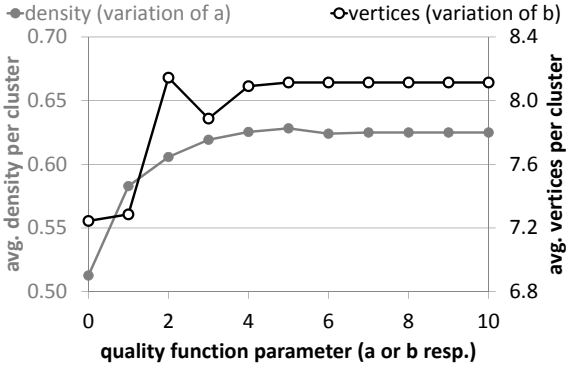


Figure 10.17: Influence of the quality function on the clustering result (Arxiv data)

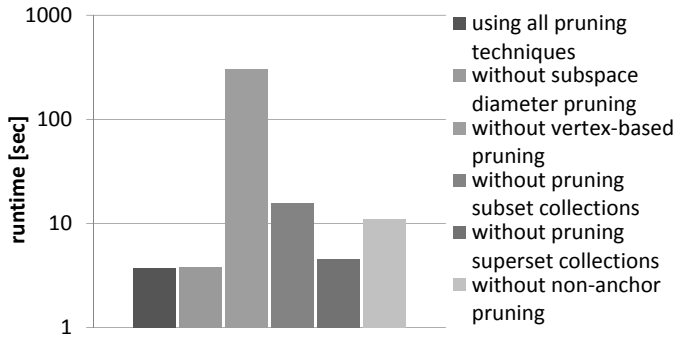


Figure 10.18: Influence of the different pruning techniques

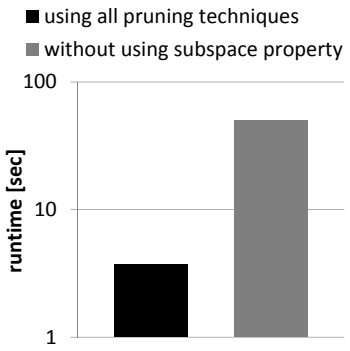


Figure 10.19: Pruning using the subspace cluster property

10.5.5 Effectiveness 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. 10.18 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 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. 10.18, using all pruning techniques simultaneously 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 in Fig. 10.18) 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 if all pruning techniques are used. 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 w.r.t. one of its subsets. The last bar shows the effect of the pruning w.r.t. non-anchor clusters. As shown in Fig. 10.18, this pruning technique also leads to a considerable efficiency gain.

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 Section 10.4.1 we developed several pruning techniques incorporating the attribute similarity into graph based pruning. In Fig. 10.19 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.

10.6 Conclusion

In this chapter 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 con-

fines 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 our integrated mining technique constantly outperforms the competing approaches in terms of efficiency and clustering quality.

In the next chapter we present an extension of our GAMER method that uses an enhanced redundancy model to further increase the clustering quality. Furthermore, to cope with the challenge of large heterogeneous data sets we present an efficient algorithm computing an approximate solution.

Chapter 11

Efficient Mining of Combined Subspace and Subgraph Clusters

Large graphs are ubiquitous in today's applications and their analysis is still a recent challenge in data mining research. Besides the mere graph structure, data sources usually provide information about single objects by high-dimensional feature vectors. To realize the full potential for knowledge extraction both information types should be considered simultaneously by using integrated mining methods. Due to the inherent complexity of such integrated models the existing methods are not efficiently executable and are hardly applicable to large graphs.

In this chapter, we develop an efficient method realizing a sound combination of subspace clustering and dense subgraph mining. To achieve this aim, we introduce a clustering model that globally optimizes the overall clustering and avoids redundancy. We prove the complexity of our model and identify the critical parts inhibiting an efficient execution. Based on this analysis, we develop the efficient and effective algorithm EDCAR that approximates the optimal clustering solution. In thorough experiments on synthetic and real world data we show that EDCAR outperforms all competing approaches in terms of runtime and simultaneously achieves high clustering qualities.

11.1 Motivation

In the past few years, real world networks have become bigger and also more numerous. Their growing availability motivated researchers and practitioners to

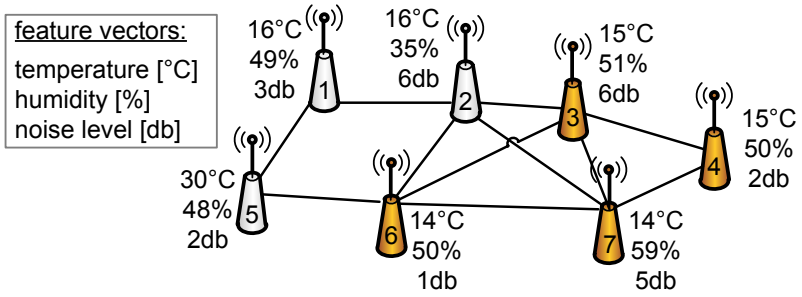


Figure 11.1: Exemplary graph with feature vectors and a combined subspace and dense subgraph cluster (located in subspace $\{\text{temperature}, \text{humidity}\}$)

analyze and use them for several purposes. One aim is to identify groups of objects, i.e. clusters, representing highly connected subsets of nodes, which is useful for, e.g., social network analysis. However, restricting the considerations only to the nodes' relations does not realize the full potential for knowledge extraction. Usually for all objects a variety of additional information is available in form of attribute data (cf. Fig. 11.1). This information allows for finding homogeneous node sets. In order to gain more informative patterns it is preferable to consider relationships together with shared characteristics.

As shown in the previous chapter, combining the paradigms of *dense subgraph mining* and *subspace clustering* leads to promising clustering results for such kind of heterogeneous data. Since clusters are identified in locally relevant subspace projections of the attribute data, the problems of (integrated) full-space clustering methods are circumvented: irrelevant dimensions do not longer obfuscate the clustering structure but are excluded. In Fig. 11.1 for example the sensors $3, 4, 6, 7$ are highly connected and they show similarity in two of their three measurements. Such integrated subspace clustering methods enable us to detect more meaningful clusters in the data, like groups of friends sharing only some product relevant attributes. However, combining the paradigms of dense subgraph mining and subspace clustering poses several efficiency challenges.

First, analyzing subspace projections is inherently hard since the number of subspaces grows exponentially in the number of attributes. Second, as shown in the previous chapter, to obtain high quality clusterings, an unbiased synthesis of both paradigms has to be conducted; none of them should be favored over the other. Thus, the clustering process has to realize a complex optimization to fairly trade off the cluster properties 'size', 'density', and 'dimensionality'. This further increases the overall complexity of the analysis. Last, often an overlap between

clusters is reasonable since objects can belong to multiple clusters when regarding different attribute subsets. Musicians of an orchestra, e.g., may share similar musical interests but probably will do sport with different persons. However, if the clustering model allows clusters to overlap, it is indispensable to avoid redundancy induced by highly overlapping clusters. As we have proven in Chapter 2 & 3 already for usual subspace clustering, redundancy elimination is highly complex.

As we have seen so far, a model for a proper combination of subspace clustering and dense subgraph mining has to handle numerous aspects. Although most of the existing approaches do not accommodate all requirements, they already have high runtime and space consumptions. Thus, an execution on large datasets (if possible at all) is not efficient.

In this chapter we deal with all the aforementioned aspects but lay special focus on the efficiency challenges described above. We start by taking the idea of GAMER to the next level. While GAMER restricts the underlying clustering model to just greedily select good clusters for the result, which does not necessarily result in the most interesting clustering, we aim at a globally optimizing clustering model inspired by our methods introduced in the first part of this thesis. Since even the models of GAMER and CoPaM [MCRE09] are rarely efficiently computable, it is not surprising that such a model, aiming at a globally optimal clustering, has a high complexity. However, in many applications the exact solution is not required but a sub-optimal solution is sufficient. We therefore analyze our model's complexity to identify the most critical parts, which inhibit an efficient execution of such integrated clustering approaches. We substitute these critical parts through highly efficient heuristics that, however, influence the clustering quality only marginally. We, thus, are able to develop a highly efficient algorithm to approximate the optimal result. It is worth mentioning that also other approaches could benefit from these heuristics. Thorough experiments demonstrate that our algorithm not only is far superior to all other approaches in terms of runtime but also shows better quality in nearly all experiments due to the globally optimizing model. The main contributions of this work are:

- We develop a novel clustering model for a result having globally maximal quality, allowing clusters to overlap in general, and avoiding redundancy
- We prove the runtime complexity of our clustering model in detail
- We introduce the efficient and effective algorithm EDCAR that approximates the optimal result

11.2 Comparison with Related Work

As discussed in the previous chapter, besides our own GAMER approach there exists just one method dealing with subspace clustering and dense subgraph mining. CoPaM's [MCRE09] combination of both paradigms, however, is not sound since important properties of the paradigms are not appropriately considered. The method solely maximizes the number of nodes, while the density of subgraphs and the subspace dimensionality only have to exceed a minimal threshold; they are thus incidental. Furthermore, CoPaM does not eliminate redundancy. Especially in the analysis of subspace projections, however, many clusters can occur which lead to an overwhelming result size. Our GAMER approach performs an unbiased synthesis of subspace clustering and dense subgraph mining by simultaneously considering the density, the size, and the dimensionality of clusters and trading off these characteristics. Furthermore, GAMER uses a redundancy model to confine the result to a manageable size. A disadvantage, however, is the simple determination of the final clustering: GAMER does not globally examine the result but simply successively adds (in a greedy manner) clusters to the result. This ranking principle was used throughout the algorithm and is the reason for the polynomial complexity (cf. Chapter 10.3.3). Due to this greedy selection, the overall clustering does not necessarily correspond to the most interesting one.

A further drawback of both methods is their high runtime and large space requirement, which prevents an application on larger datasets. With our novel algorithm we try to solve this drawback by efficiently determining an approximate solution of the optimal clustering result.

11.3 Quality-Optimal Clustering

Our EDCAR method (**E**fficient **D**etermination of **C**lusters regarding **A**ttributes and **R**elationships) realizes a novel clustering model. The model is based on the cluster definition introduced in our GAMER approach. The input of our model is a vertex-labeled graph $G = (V, E, l)$ with vertices V, edges $E \subseteq V \times V$ and a labeling function $l : V \rightarrow \mathbb{R}^d$ where $Dim = \{1, \ldots, d\}$ is the set of dimensions. We assume an undirected graph without self-loops, i.e. $(v, u) \in E \Leftrightarrow (u, v) \in E$ and $(u, u) \notin E$. We use $l(O) = \{l(o) \mid o \in O\}$ to denote the set of vectors that is associated to the set of vertices $O \subseteq V$.

In Section 11.3.1 we briefly describe the cluster definition taken over from GAMER and we focus on our cluster selection procedure to get the most interesting clustering. In Section 11.3.2 we analyze the complexity of our model.

11.3.1 Clustering model

Our method combines objectives from subspace clustering and dense subgraph mining. As our cluster definition we use the twofold cluster model as introduced in Definition 10.3, i.e. a set of objects $O \subseteq V$ and a set of dimensions $S \subseteq Dim$ is a valid twofold clusters if O is a quasi-clique in the given graph and (O, S) is a subspace cluster. For example in Fig. 11.2 (choosing $w = 0.5$, $n_{min} = 3$, $\gamma_{min} = 0.4$ and $s_{min} = 2$) the vertex set $C_1 = \{v_1, v_2, v_4, v_5, v_6, v_7\}$ is a valid cluster with the relevant dimensions 2 and 3 (marked in orange). The set $C_2 = \{v_2, v_3, v_7\}$ is a cluster with the relevant dimensions 1 and 4 (marked in blue).

Based on the definition of twofold clusters (cf. Def. 10.3), the number of node sets fulfilling this definition is potentially very large and probably many clusters will overlap. Since these redundant clusters are not beneficial but obstructing for the user, they should be excluded from the result. To identify a redundant cluster C w.r.t. another cluster C', we use the binary redundancy relation as introduced in the GAMER model:

Definition 11.1 (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 $C = (O, S), C' = (O', S')$:

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

Note that a redundant cluster has to fulfill multiple conditions: First, the cluster C should be less interesting than the cluster C'; otherwise one would prefer C. This interestingness is modeled by the quality function $Q(C)$ and can be easily adapted by the user. Second, the structural information of the corresponding clusters has to be similar, i.e. they have to share a large portion of their vertices and their dimensions. This is realized by the last two conditions of the above defined redundancy relation.

Cluster selection based on global optimization Due to the redundancy problem, it is usually not beneficial to output the whole set $Clusters$ of all twofold

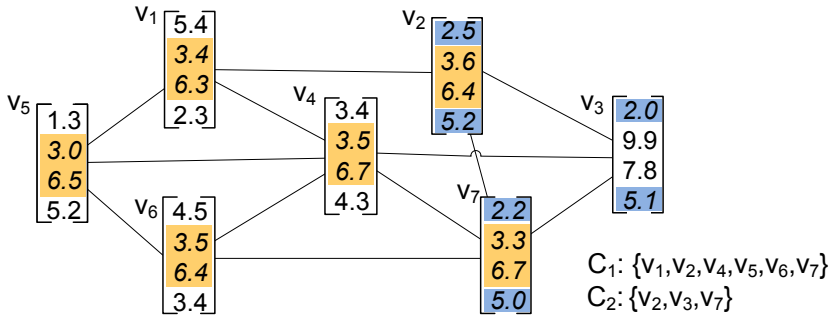


Figure 11.2: Exemplary twofold clusters

clusters. In contrast, we select the final clustering $Result$ as a subset $Result \subseteq Clusters$. Our goal in selecting the final clustering is to extract the most interesting clustering which potentially contains overlapping clusters but at the same time minimizes the redundancy in the final output.

As a first property, our $Result$ must not contain clusters that are redundant to each other, i.e. it has to be *redundancy-free*: for each cluster $C_i \in Result$ there must not exist another cluster $C_j \in Result$ with $C_i \prec_{red} C_j$. Obviously, an empty result fulfills this property but it is not useful. Thus, we also want to select the most interesting clustering.

While the GAMER method greedily selects clusters according to their quality, we perform a more sophisticated selection in our approach. Instead of deciding locally which cluster to select next for the result, we perform a global optimization to get the most interesting clustering. Thus, we do not prefer the selection of *single* interesting *clusters*, since that carries the risk of selecting only uninteresting clusters afterwards, but we select the *overall* most interesting *clustering*. Correspondingly, our $Result$ has to fulfill the *maximum quality sum property*, which means that the sum of the clusters' qualities in $Result$ has to be maximal compared to all other possible clusterings. Formally, the quality-optimal clustering is defined as follows:

Definition 11.2 (Quality-optimal clustering) *Given the set of all twofold clusters $Clusters$, the quality-optimal clustering $Result \subseteq Clusters$ fulfills*

- *(redundancy-free)* $\neg\exists C_i, C_j \in Result : C_i \prec_{red} C_j$
- *(maximum quality sum)* $\neg\exists Res' \subseteq Clusters$: *Res' fulfills the redundancy-free property and* $\sum_{C_i \in Res'} Q(C_i) > \sum_{C_i \in Result} Q(C_i)$

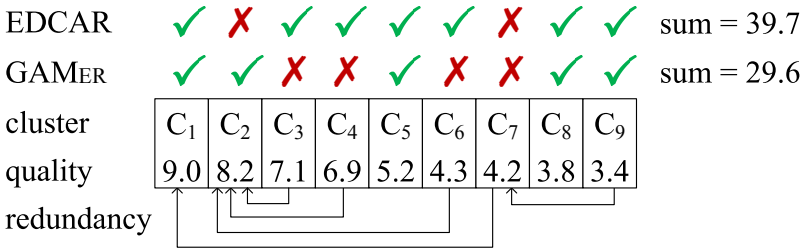


Figure 11.3: Selecting the final clustering: global optimization in EDCAR vs. greedy selection in GAMER

Fig. 11.3 shows an example for the final clusterings of GAMER and EDCAR. In the example, nine clusters, their corresponding quality values, and the redundancy relation are illustrated. The quality sums of the determined clusterings are depicted on the right of the figure. GAMER selects the cluster C_2 since it is not redundant w.r.t. any other cluster. A greedy selection according to the quality values is performed. In EDCAR, cluster C_2 is not selected for the final clustering. While C_2 has a high quality itself, by including it we would have to exclude the clusters C_3, C_4 and C_6. However, by including these clusters and excluding C_2 we obtain a final clustering of higher quality (39.7 vs. 29.6).

As the example illustrates, EDCAR optimizes the interestingness of the overall clustering, which can yield better results but is computationally more challenging.

11.3.2 Complexity analysis

In this section we analyze the complexity of our clustering model. First, the overall complexity of our model, i.e. generating the twofold clusters *and* selecting the quality-optimal clustering, is #P-hard (like the GAMER model). We denote this problem as $OVERALL$. Second, even if the set of twofold clusters $Clusters$ is given, selecting the quality-optimal clustering $Result \subseteq Clusters$ according to Definition 11.2 is NP-complete w.r.t. the input size. This is in contrast to the GAMER model, where selecting the clustering can be done in polynomial time. We denote this subproblem of our model as $SELECT$.

Theorem 11.1 *Given a vertex-labeled graph $G = (V, E, l)$, determining the quality-optimal clustering according to Def. 11.2 is #P-hard w.r.t. 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 $MAXCLIQUE$ [GJ79] 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 11.1
Input Mapping: *The graph G is taken as it is. We choose $\gamma_{min} = 1$, $n_{min} = k$, $s_{min} = 0$, $r_{obj} = 1$, $r_{dim} = 0$, $a = c = 0$, and $b > \log_{\frac{x}{x-1}}(2^x - 2)$ with $x = \max\{2, |V|\}$.*

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 proper subsets of clusters induce redundancy, i.e.*

$$C = (O, S) \prec_{red} C' = (O', S')$$

$$\Leftrightarrow Q(C) < Q(C') \wedge \frac{|O \cap O'|}{|O|} \geq 1 \wedge \frac{|S \cap S'|}{|S|} \geq 0$$

$$\Leftrightarrow |O|^b < |O'|^b \wedge O \subseteq O' \Leftrightarrow O \subset O'$$

Consequently, for a maximal cluster C_m there exists no C with $C_m \prec_{red} C$.

(3): *To ensure that only maximal clusters (w.r.t. their objects) are selected, any cluster $C = (O, S)$ must get a higher quality than the quality-sum of all of its potential (non-empty) subsets, i.e.*

$$Q(C) = |O|^b > \sum_{O' \subset O, O' \neq \emptyset} |O'|^b$$

We infer

$$\sum_{O' \subset O, O' \neq \emptyset} |O'|^b \leq \sum_{O' \subset O, O' \neq \emptyset} (|O| - 1)^b = (2^{|O|} - 2)(|O| - 1)^b$$

Now, we have to prove for each (non-empty) cluster $C = (O, S)$ the property

$$|O|^b > (2^{|O|} - 2)(|O| - 1)^b \quad \Leftrightarrow \quad \begin{cases} b > \log_{\frac{|O|}{|O|-1}}(2^{|O|} - 2) & |O| > 1 \\ b \in \mathbb{R} & |O| = 1 \end{cases}$$

Since $\log_{\frac{|O|}{|O|-1}}(2^{|O|}-2)$ *is monotonically increasing in* $|O|$*, the inequality is obviously true for all clusters by choosing* b *as follows:*

$$b > \log_{\frac{x}{x-1}}(2^x - 2) \text{ with } x = \max\{2, |V|\}$$

(4): $Result$ *contains all maximal clusters* C_m *w.r.t. their size.* Proof by contradiction: *Assumption:* C_m *is maximal but* $C_m = (O, .) \notin Result \Rightarrow$ *it exists a maximal subset* $M \subseteq Result$ *with*

$$\forall C' = (O', .) \in M : \; C' \prec_{red} C_m \vee C_m \prec_{red} C'$$

$$\overset{(2)}{\Leftrightarrow} \forall C' = (O', .) \in M : \; C' \prec_{red} C_m$$

$$\Leftrightarrow \forall C' = (O', .) \in M : \; O' \subset O$$

Thus, by removing M *from* $Result$ *and adding* C_m *we will get a novel redundancy-free clustering* Res'*. However, based on (3) we know that* $Q(C_m) > \sum_{C \in M} Q(C)$*. Thus,* Res' *has higher quality and* $Result$ *cannot be the optimal clustering.*

The assumption was wrong $\Rightarrow C_m \in Result$

(5): *Based on (1), (2) and (4),* $Result$ *contains all and only the maximal clusters, which correspond to the maximal cliques. Thus,* $OVERALL$ *generates a valid solution for* $MAXCLIQUE$*.* □

Now we prove the complexity of the cluster selection process. While in the GAMER method the complexity was polynomial w.r.t. the number of clusters, our quality-optimal clustering is NP-complete w.r.t. the input size.

Theorem 11.2 *Given a set of twofold clusters* $Clusters$*, selecting the quality-optimal clustering according to Def. 11.2 is NP-complete w.r.t. the number of clusters.*

We show the complexity of this subproblem $SELECT$ by a polynomial reduction to the maximum weighted set packing problem ($MWSP$) and vice versa, i.e. we prove

$$MWSP \leq_P SELECT \leq_P MWSP$$

The input for the maximum weighted set packing problem consists of a base set of objects $U = \{u_1, \ldots, u_n\}$ and a set of sets $T = \{U_1, \ldots, U_l\}$ with $U_1, \ldots, U_l \subseteq U$ where each set U_i is annotated with a weight $w_i \in \mathbb{Q}$. The task is now to choose

a subset $M \subseteq T$ of pairwise disjoint sets such that $\sum_{U_i \in M} w_i$ is maximal. This problem is known to be NP-complete and is hard to approximate [Pas97].

We first show that $SELECT$ can be used to solve the $MWSP$ problem, i.e. $MWSP \leq_P SELECT$.

Proof 11.2
W.l.o.g. we assume $w_i \in \mathbb{N}$ (rational numbers can be scaled by a constant factor to obtain natural numbers; this scaling obviously does not influence the optimal solution).

Input Mapping: *The input of $MWSP$ has to be mapped to an input for $SELECT$. Thus, each U_i and w_i is mapped to a cluster $C_i = (U_i, \{1, \ldots, l^2 \cdot w_i + i\})$ and we choose $r_{obj} = \frac{1}{n}$, $r_{dim} = 0$, $a = 0$, $b = 0$, $c = 1$. Obviously, this is a polynomial reduction.*

Output Mapping: *The final clustering $Result$ obtained by $SELECT$ represents the $MWSP$ solution $U_i \in M \Leftrightarrow C_i \in Result$. We prove this fact in several steps:*

(1): *We show that each cluster has a different quality:*

- *Case $w_i = w_j$ (for $i \neq j$):*

$$l^2 \cdot w_i + i = l^2 \cdot w_j + i \neq l^2 \cdot w_j + j$$

  $\Rightarrow$ *the subspaces of C_i and C_j have different sizes*
  $\Rightarrow Q(C_i) \neq Q(C_j)$.

- *Case $w_i > w_j$:*

$$w_i - w_j \geq 1 \quad \Leftrightarrow \quad l^2 \cdot w_i - l^2 \cdot w_j \geq l^2$$

$$\Leftrightarrow (l^2 \cdot w_i + i) - (l^2 \cdot w_j + j) \geq l^2 + i - j \geq l^2 + 1 - l \geq 1$$

$$\Rightarrow l^2 \cdot w_i + i > l^2 \cdot w_j + j$$

  $\Rightarrow$ *the subspaces of C_i and C_j have different sizes*
  $\Rightarrow Q(C_i) \neq Q(C_j)$.

- *Case $w_i < w_j$ analogously.*

(2): *Redundancy holds iff object sets are not disjoint:*
As the redundancy parameters are set to $r_{obj} = \frac{1}{n}$, $r_{dim} = 0$ *it holds that*

$$C_i \prec_{red} C_j \quad \Leftrightarrow \quad Q(C_i) < Q(C_j) \wedge \frac{|O_i \cap O_j|}{|O_i|} \geq \frac{1}{n} \wedge \frac{|S_i \cap S_j|}{|S_i|} \geq 0$$

$$\overset{|O_i| \leq n}{\Leftrightarrow} Q(C_i) < Q(C_j) \wedge |O_i \cap O_j| \geq 1$$

Since all cluster qualities are different (see (1)) we get

$$C_i \prec_{red} C_j \vee C_j \prec_{red} C_i$$

$$\Leftrightarrow (Q(C_i) < Q(C_j) \vee Q(C_j) < Q(C_i)) \wedge |O_i \cap O_j| \geq 1$$

$$\overset{(1)}{\Leftrightarrow} O_i \cap O_j \neq \emptyset$$

(3): *A redundancy-free clustering contains only pairwise disjoint objects sets, since it holds*

$$\neg \exists C_i, C_j \in Result: \; C_i \prec_{red} C_j \vee C_j \prec_{red} C_i$$

$$\Leftrightarrow \neg \exists U_i, U_j \in M: \; U_i \cap U_j \neq \emptyset$$

(4): *For a clustering* $Result$ *with maximal quality (and* $\forall Res' \subseteq Clusters$*) the following bounds apply:*

$$\sum_{C_i \in Res'} Q(C_i) \leq \sum_{C_i \in Result} Q(C_i)$$

We infer

$$l^2 \sum_{C_i \in Res'} w_i \leq \sum_{C_i \in Res'} l^2 \cdot w_i + i = \sum_{C_i \in Res'} Q(C_i) \leq \sum_{C_i \in Result} Q(C_i)$$

$$= l^2 \sum_{C_i \in Result} w_i + \sum_{C_i \in Result} i \leq l^2 \sum_{C_i \in Result} w_i + \frac{(l+1)l}{2}$$

W.l.o.g. we have $l > 1$*, thus dividing by* l^2 *we get*

$$\sum_{C_i \in Res'} w_i \leq \sum_{C_i \in Result} w_i + \frac{(l+1)}{2 \cdot l} < \sum_{C_i \in Result} w_i + 1$$

(5): *A clustering $Result$ with maximal quality sum corresponds to a set M with maximal overall weight:*

Assumption: M has not maximal weight, i.e. it exists M' with $\sum_{U_i \in M'} w_i > \sum_{U_i \in M} w_i$. Since all the weights are natural numbers, the difference in the overall weights has to be at least 1

$$\Rightarrow \sum_{U_i \in M'} w_i - \sum_{U_i \in M} w_i \geq 1 \Rightarrow \sum_{U_i \in M'} w_i \geq \sum_{U_i \in M} w_i + 1$$

However, according to (4) we know

$$\sum_{C_i \in Res'} w_i < \sum_{C_i \in Result} w_i + 1$$

Due to the correspondence between M/M' and $Result/Res'$, the assumption has to be wrong. Thus, M is a set with maximal weight.

(6): *Based on (3) and (5), $Result$ contains those pairwise disjoint clusters that have overall maximal weight. Thus, $SELECT$ generates a valid solution for $MWSP$.* □

We now show $SELECT \leq_P MWSP$, i.e. $MWSP$ can be used to solve $SELECT$.

Proof 11.3
Input Mapping: *The base set U is determined by*

$$U = \{i, \{i, j\} \mid C_i, C_j \in Clusters\}$$

Each cluster $C_i \in Clusters$ is mapped to a set

$$U_i = \{i\} \cup \{\{i, j\} \in U \mid C_i \prec_{red} C_j \vee C_j \prec_{red} C_i\}$$

The set U_i represents the cluster itself and its redundant counterparts. The weights correspond to the clusters quality, i.e. $w_i = Q(C_i)$. This is a polynomial reduction since the redundancy relation $\prec_{red}$ can be determined in polynomial time.

Output Mapping: *The result M obtained by $MWSP$ represents the clustering $C_i \in Result \Leftrightarrow U_i \in M$.*

(1): *Obviously, it holds*

$$C_i \prec_{red} C_j \vee C_j \prec_{red} C_i \Leftrightarrow U_i \cap U_j = \{\{i, j\}\}$$

(2): *The resulting clustering is redundancy-free since*

$$\forall U_i, U_j \in M : \ U_i \cap U_j = \emptyset \Rightarrow \neg\exists U_i, U_j \in M : \ |U_i \cap U_j| > 0$$

$$\Rightarrow \neg\exists C_i, C_j \in Result : \ C_i \prec_{red} C_j \vee C_j \prec_{red} C_i$$

(3): *The resulting clustering has maximum quality since*

$$\sum_{U_i \in M} w_i = \sum_{C_i \in Result} Q(C_i)$$

is maximal
(4): *Based on (2) and (3),* M *represents a redundancy-free clustering with maximal quality. Thus,* $MWSP$ *generates a valid solution for* $SELECT$. □

Conclusions From Theorem 11.1 (and similarly stated in Corollary 10.2) we can infer that the number of twofold clusters which provide the input for the cluster selection process can be exponential in the number of vertices. Thus, we can identify *two parts* that, especially in combination, *prevent an efficient determination* of the optimal clustering: the tremendous amount of clusters used as candidates for the optimal result and the complexity of the selection process for a final subset of these clusters.

11.4 The EDCAR Algorithm

As shown, efficiently determining the optimal clustering is not possible. Thus, we develop the heuristic algorithm EDCAR to ensure an efficient execution by approximating the optimal result. We have to tackle the two important challenges: First, the number of result candidates has to be small. We cannot use the whole set $Clusters$ of exponentially many candidates as the input for the selection procedure. Second, we have to resolve the NP-hardness of the selection process itself. In Section 11.4.1 we show how the candidates for the result are reduced to a

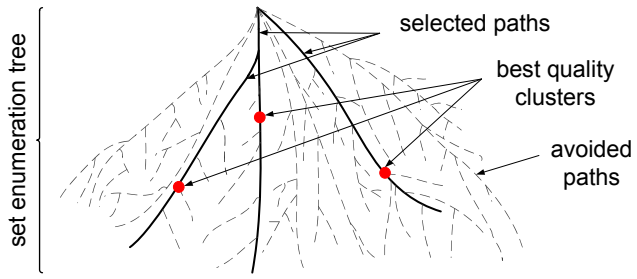


Figure 11.4: Generating a reduced number of candidates

manageable size and in Section 11.4.2 we approximate the selection process. Section 11.4.3 describes the overall processing, which introduces a method to increase the clustering quality of the final result.

11.4.1 Reduce the number of result candidates

This first phase generates the cluster candidates among which the subsequent process chooses the final clustering. The goal is to *efficiently* determine a set of twofold clusters that is of *manageable size* and of *high quality*. Consequently, we do neither want to analyze all sets of vertices nor want to include unnecessarily many candidates for the NP-complete selection process.

As in our GAMER algorithm, we use the set enumeration tree to analyze sets of vertices whether they are valid twofold clusters (cf. Chapter 10.4.1). To recap the idea, we refer to Fig. 10.4: Each node of the set enumeration tree represents a set of vertices $O \subseteq V$. All nodes together represent the powerset of V, which is the set of all potential clusters. A child node O' extends its parent node O through one of the vertices stored in the candidate set $cand_O$. Thus, the subtree of a node O represents all potential clusters X with $O \subset X \subseteq O \cup cand_O$. By pruning a vertex v from the candidate set of a node O, the search space can be reduced. If we were able to remove e.g. the vertex v_3 from the set $cand_{\{v_1\}}$, the highlighted subsets in Fig. 10.4 would disqualify themselves as clusters without further analysis. EDCAR employs the pruning methods introduced in the Chapter 10.4.1 to prune nodes that lead to invalid clusters.

As stated before, we want to avoid to analyze *each* node along *each* (non-pruned) path in the set enumeration tree since this could lead to an exponential number of twofold clusters which are used as candidates for the final clustering. To reduce the number of candidates we implement two different strategies. In the

first step, we avoid analyzing all paths of the tree by *systematically determining single paths* along which interesting clusters can be expected. By selecting a polynomial number of paths we will also only get a polynomial number of candidates since the number of clusters per path is restricted by its length. This path selection is illustrated in Fig. 11.4 by the solid lines. Even though this method reduces the number of clusters considerably, this set is still unnecessarily large and will be reduced in a second step. Since along each selected path in the set enumeration tree the object sets successively grow by one vertex (cf. Fig. 10.4), the analyzed sets and accordingly the subspaces are very similar. Thus, the clusters along a single path are most likely redundant to each other. Using all these clusters as the input for the selection step is needless since most clusters will be discarded anyway. Based on the definition of our redundancy relation we know that clusters with high quality are preferred. Thus, instead of using all clusters, we further reduce the number of candidates by selecting along each path just the cluster with the highest quality. This path reduction is indicated by the dots in Fig. 11.4. Overall, we realize by our two strategies that only few candidates are used as the input for the cluster selection step. Thus, this step considerably contributes to the efficiency of our algorithm. In the remaining of this section, we present more details on our path selection technique.

GRASP for efficient path selection. The question remains open how to systematically (and efficiently) achieve a selection of interesting paths. For this purpose, we adapt the GRASP principle (Greedy Randomized Adaptive Search Procedure) [PR02, Res09, RR10], which is, in our scenario, motivated by the following observations: Naively, one could randomly determine a path, i.e. one extends a (partial) path ending in the node O of the set enumeration tree by randomly expanding it with a vertex $v \in cand_O$. This, however, does not assure to generate high quality clusters. Alternatively, one could decide at node O which successor v (potentially) leads to a good cluster and one descends in the subtree with the highest potential. Thus, we formally need an (efficiently computable) function $g(v|O)$ to estimate the potential of each node $v \in cand_O$ depending on the current set O. The definition of the function $g(v|O)$ will be derived in the next subsection. This approach corresponds to a greedy construction of the path since previous decisions which subtrees to analyze are not revised. The huge advantage of this greedy method is that the graph structure and the cluster definition can be incorporated into the

Algorithm 11.1: generateCandidates()

```
1  Cands ← {}
2  for 1 ... maxClusterPerIter do
3      α ← rand([0 ... 1]) // trade off greedy & randomized
4      O ← {} // root of set enumeration tree
5      while true do // generate path P
6          determine cand_O and prune (cf. Chapter 10.4.1)
7          if cand_O = ∅ then break;
8          determine RCL_O as in Equation 11.1
9          v ← rand(RCL_O)
10         // extend path P by v
11         O ← O ∪ {v}
12     select cluster C+ along path P with highest quality
13     Cands ← Cands ∪ {C+}
14 return Cands
```

estimation function g to rate the potential of the path. Thus, it corresponds to an *informed search* and high quality clusters can be expected. Disadvantageous is the risk of reaching only local maxima and generating always very similar paths (if a deterministic estimation is used). These problems do not hold for the randomized approach, which is able to generate a *diversity of paths* in an *uninformed* fashion.

To exploit the advantages of both methods, the GRASP principle, which acts as a metaheuristic to combine them, is used in our method. Several studies show that this principle often leads to optimal or nearly optimal solutions [PR02]. Thus, according to the GRASP principle, we first construct a restricted candidate list (RCL) corresponding to a *set* of potentially meaningful vertices for expanding the path. Afterwards, we randomly select one vertex $v \in RCL_O$ to descend into a subtree. Formally,

$$RCL_O = \{v \in cand_O \mid g(v|O) \geq s_{min} + \alpha \cdot (s_{max} - s_{min})\} \tag{11.1}$$

with $s_{min} = \min_{v \in cand_O} \{g(v|O)\}$ and $s_{max} = \max_{v \in cand_O} \{g(v|O)\}$. Thus, the vertices with the highest potential are included in the RCL. By choosing $\alpha = 1$ we can simulate the greedy approach whereas $\alpha = 0$ corresponds to a completely randomized selection. The determination of a single path is shown in Algorithm 11.1, line 5 to 11: In line 6 we determine the candidate set $cand_O$ for the current vertex set O, which is then reduced using the pruning techniques introduced in our GAMER approach (cf.

Chapter 10.4.1). These techniques only exclude vertices from $cand_O$ that can not lead to twofold clusters. Thus, for the path extension only the remaining vertices need to be considered. Next we determine the RCL as described above and add a randomly selected node to O. This procedure is repeated until $cand_O$=$\emptyset$, i.e. no more nodes can be added to the path.

As depicted in Fig. 11.4 we want to descend in several paths. The number of processed paths is given by the parameter *maxClusterPerIter*. This is done in line 2 and we use a randomly determined α to trade off the two GRASP principles for each novel path, leading to more stable results [RR10]. Overall, we efficiently generate different paths that contain high quality clusters based on the estimated potential.

Potential of paths. At last, we have to determine the potential of a subtree. In our case, the potential directly depends on the quality of the expected clusters (cf. Def. 10.4). We therefore want to use the quality as our estimation function $g(v|O)$. If efficiency was not required, one could use $g'(v|O) = \gamma(O \cup \{v\})^a \cdot |O \cup \{v\}|^b \cdot S(O \cup \{v\})^c$ where $S(X)$ denotes the subspace of the corresponding vertex set. In this case, the density, the size, and the dimensionality of the potential cluster $O \cup \{v\}$ are exactly calculated and thus also the quality. One prefers to expand the path by the most interesting vertices. However, efficiency is crucial in our case. Thus, $g(v|O)$ has to be efficiently computable. While the size and the subspace can be efficiently determined, the exact density is computationally expensive. Keep in mind that $g(v|O)$ has to be evaluated for each $v \in cand_O$. Therefore, we approximate the density $\gamma(O \cup \{v\})$ of the potential cluster $O \cup \{v\}$.

Theorem 11.3 *The density of the set $O \cup \{v\}$ can be lower bounded by*

$$\overline{\gamma}(O, v) := \frac{\min\{\min_{o \in O} deg^O(o), deg^O(v)\}}{|O|} \leq \gamma(O \cup \{v\})$$

Proof 11.4
$$\overline{\gamma}(O, v) = \frac{\min\{\min_{o \in O} deg^O(o), deg^O(v)\}}{|O|} = \frac{\min_{o \in O \cup \{v\}} deg^O(o)}{|O|}$$
$$\leq \frac{\min_{o \in O \cup \{v\}} deg^{O \cup \{v\}}(o)}{|O \cup \{v\}| - 1} = \gamma(O \cup \{v\}) \qquad \square$$

The density approximation $\overline{\gamma}(O, v)$ can be efficiently computed for different vertices v because it is mostly independent of v. We only have to compute the term $deg^O(v)$. Our overall estimation function is

$$g(v|O) = \overline{\gamma}(O, v)^a \cdot (|O| + 1)^b \cdot S(O \cup \{v\})^c$$

11.4.2 GRASP for efficient clustering selection

In the previous section we reduced the number of candidates used as the input for the cluster selection step. Now we want to approximate the optimal clustering according to Definition 11.2 itself since its determination is NP-hard w.r.t. a given set of candidates $Cands$. At first we will relax Definition 11.2 by only demanding the resulting clustering Res to be redundancy-free and maximal:
$(\neg\exists C, C' \in Res : C \prec_{red} C') \wedge (\forall C \in Cands \backslash Res : \exists C' \in Res : C \prec_{red} C' \vee C' \prec_{red} C)$

To still guarantee a high quality of the clustering we again use the GRASP principle. Previously, we used GRASP to extend the path by further vertices, now we have to extend a (preliminary) result set Res by further clusters $C \in Cands$. Thus, we have to define an estimation function $h(C|Res)$ to assess the potential of adding C to Res. Since we want to increase the quality sum as much as possible, we directly use the quality of clusters: $h(C|Res) = Q(C)$. This value is already given at this time. Thus, no additional computation has to be done. The pseudo code for selecting the subset is given in Algorithm 11.2. Based on the possible clusters $Cands$ the RCL is determined and one cluster is randomly selected to expand the result set (line 5-7). Note that our model requires the redundancy-free property. Therefore, we are able to remove in line 8 all clusters that induce such a redundancy. These clusters can no longer be added to Res and hence they are removed from $Cands$.

Algorithm 11.2: selectClustering(set of clusters $Cands$)

```
1  Res ← {} // (preliminary) result
2  α ← rand([0...1])
3  while Cands ≠ ∅ do
4  |   s_min = min_{C∈Cands} h(C|Res), s_max = max_{C∈Cands} h(C|Res)
5  |   RCL={s ∈ Cands | h(C|Res) ≥ s_min + α(s_max − s_min)}
6  |   C ← rand(RCL)
7  |   Res ← Res ∪ {C}
8  |   Cands ← Cands\{C ∈ Cands | ∃C' ∈ Res : C ≺_red C' ∨ C' ≺_red C}
9  // improve clustering by local search
10 for NewRes ∈ Neighborhood(Res) do
11 |   if Q(NewRes) > Q(Res) then // higher quality sum
12 |   |   Res ← NewRes
13 |   |   goto line 9 // efficient first-improving strategy
14 return Res
```

To further increase the overall quality, we conduct in line 9-13 a local search on the set of valid clusterings. The idea is to replace a cluster $C \in Res$ by a set of not yet selected clusters New_C to get the potentially better clustering $Res \backslash \{C\} \cup New_C$. The set New_C is built by collecting clusters from $Cands \backslash Res$, in decreasing order w.r.t. their quality values, as long as the redundancy-free property of the overall result $Res \backslash \{C\} \cup New_C$ is not violated. Thus, an efficient greedy approach can be used to generate the set New_C. Formally, for each cluster $X \in Cands \backslash Res$ *not* selected for New_C it holds:

$$X \notin New_C \Leftrightarrow \exists C' \in New_C : \ (X \prec_{red} C') \\ \vee \exists C' \in Res \backslash \{C\} : (X \prec_{red} C' \vee C' \prec_{red} X)$$

Note that there is no term $(\exists C' \in New_C : \ C' \prec_{red} X)$ since then $Q(X) > Q(C')$ would hold and thus C' would not be included in New_C. The overall neighborhood of a clustering Res is the whole set of such generated alternatives

$$Neighborhood(Res) = \{Res \backslash \{C\} \cup New_C \mid \forall C \in Res\}$$

If a clustering of this neighborhood with a higher quality sum is identified (line 11), we replace the current solution and we repeat the local search based on this novel result. If no better clustering exists, we have reached a local maximum. Thus, the cluster selection in this iteration is finished.

11.4.3 Overall processing scheme

The two phases of our method, generating a small number of candidates and selecting the resulting clustering based on these candidates, lead to an overall efficient execution. However, by just executing the two phases once, we cannot ensure a high quality of the final clustering. Selecting just the one cluster with the highest quality along each path, on the one hand, accounts for high efficiency but, on the other hand, prevents lower quality clusters from getting the chance to be selected for the result. Nevertheless, also clusters with lower qualities can contribute to the overall result, as C_9 in Fig. 11.3. To give these low-quality but valuable clusters the chance to be considered as result candidates, we repeat both phases recurrently (cf. Algorithm 11.3). While improving the overall quality of the result, this will also maintain the efficiency.

In the first iteration no information is given ($Res = \emptyset$) and we select the clusters with highest quality along each path. However, after the first iteration, we

Algorithm 11.3: EDCAR algorithm

```
1 Res ← {} // preliminary result
2 do
3 |   Cands ← generateCandidates()  // non-red. to Res
4 |   Tmp ← Res ∪ Cands              // expansion
5 |   Res ← selectClustering(Tmp)    // contraction
6 while Cands ≠ ∅
7 return Res
```

already have a preliminary clustering result Res. This result set is used to avoid considering redundant candidates, which otherwise would probably have to be rejected later by the complex selection process. Prioritizing non-redundant clusters is meaningful since they have bigger chances to manage an improvement of the current result. We thus block redundant parts of a path and select the most interesting cluster among the remaining non-redundant ones as additional candidate. Overall, we generate in each iteration only candidates fulfilling

$$\forall C \in Cands: \ \neg\exists C' \in Res: \ C \prec_{red} C'$$

It is very likely that some of these clusters can be added to the final clustering. Thus, we perform the clustering selection phase on the enriched set $Cands \cup Res$ to get the novel preliminary result $Res^* \subseteq Cands \cup Res$. Keep in mind that the redundancy is not ensured the other way around, i.e. $\exists C \in Cands: \ \exists C' \in Res: \ C' \prec_{red} C$ is possible. In this case $C \in Cands$ has a higher quality than C' and hence a selection of C is preferable. For the same reason, we have to perform the final selection on the whole set $Cands \cup Res$ since the redundancy-free property is not ensured before.

Overall, our processing interweaves the generation and the selection of clusters by cyclically invoking both phases. The set of considered clusters is alternately expanded by further candidates and contracted by the selection step. As further advantage, the method automatically terminates if no further non-redundant candidates can be identified in one iteration. The general EDCAR processing is given in Algorithm 11.3.

By successively generating (high-quality) clusters, we ensure the effectiveness of EDCAR. By focusing on generating a small number of (non-redundant) candidates in each step and by approximating the clustering selection itself we ensure the efficiency.

11.5 Experimental Analysis

Setup. We compare EDCAR with our GAMER method and with CoPaM [MCRE09]; both consider subspaces and dense subgraphs. As a further competitor we extend Cocain [ZWZK06] to handle our data. Originally, this method identifies object sets corresponding to quasi-cliques in several graphs. Since we analyze single vertex labeled graphs, we extract one graph for each dimension and retain only those edges of the original graph connecting nodes with similar attribute values in the particular dimension. Thereby, we simulate subspace clustering with our Cocain°. We use synthetic data, by default with 80 clusters, each with 15 nodes, a density of 0.6 and 5-10 relevant dimensions out of 20 dimensions. 6% of the clusters nodes overlap. Furthermore, we use public available real world data including *genes*[1], *patent data*[2], the *Arxiv database*[3] and the *DBLP database*[4]. We provide all datasets with descriptions as well as executables and parameter settings on our website.

Efficiency is measured by the approaches' runtime. For comparability all experiments were conducted on Opteron 2.3GHz CPUs using Java6 64 bit. Methods that did not finish within two days were aborted. Clustering quality is calculated via the F1 value as also done in the previous chapters. 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.

Scalability w.r.t. database size. The primary objective of our method is its efficient execution. If the quality of the resulting clusterings, however, is low, a high performance will be futile. Thus, besides the runtimes we also depict the algorithms' clustering qualities. We start our analysis by varying the database size, i.e. we increase the number of vertices in the graph by increasing the number of clusters and keeping the number of objects per cluster fixed. As depicted in Figure 11.5 (left), EDCAR is several orders of magnitude faster than all competing approaches (note the logarithmic scale on both axes). EDCAR is the only method applicable on large data sets. Especially CoPaM is no longer executable for these settings since its limited redundancy model leads to an impracticably large amount of clusters. GAMER scales worse than EDCAR too because it has to analyze the complete set

[1] http://thebiogrid.org, http://genomebiology.com/2005/6/3/R22
[2] http://www.nber.org/patents/
[3] http://www.cs.cornell.edu/projects/kddcup/datasets.html
[4] http://dblp.uni-trier.de

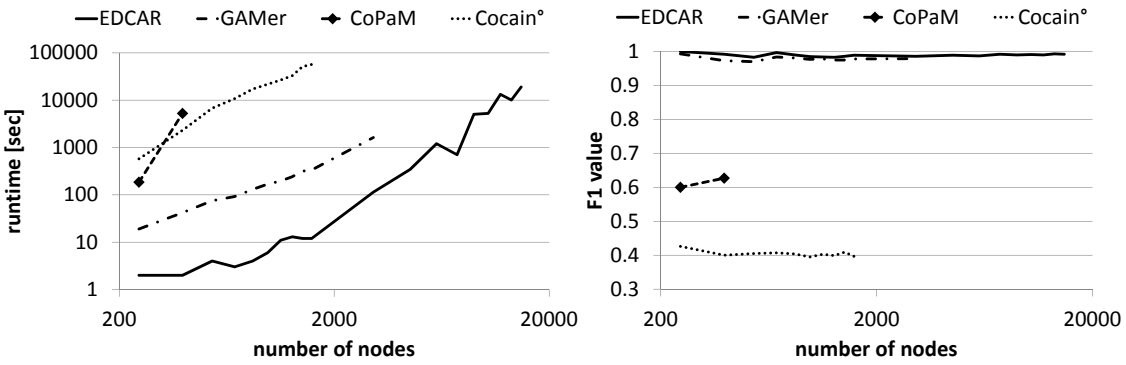


Figure 11.5: Scalability and quality w.r.t. database size

of all twofold clusters, leading to a high runtime. Although EDCAR uses an even more complex clustering model, the runtime is lower since we systematically generate and select only the most interesting clusters. A huge amount of clusters is not examined at all.

Besides EDCAR's high efficiency, we observe in Figure 11.5 (right) its high effectiveness. Despite the used approximations, the quality of EDCAR is similar or even higher than that of GAMER. We are able to detect the hidden clusters in the data. The remaining approaches CoPaM and Cocain° achieve only low qualities, as already shown in the previous chapter of this thesis.

This experiment has shown that EDCAR is also applicable on large datasets. Though, for the following experiments we chose medium-sized datasets to enable a comparison with the other algorithms.

Scalability w.r.t. cluster size. Next we increase the database size by keeping the number of clusters fix but increasing the number of vertices per cluster. This setting is more challenging since longer paths within the set enumeration tree has to be generated due to the cluster sizes. In Figure 11.6 (left) we observe a very good scalability of EDCAR: the increase in runtime is only slow. EDCAR generates only few paths within the set enumeration tree and extending a path by a further node is efficiently done by the GRASP principle. All competing approaches show heavily increasing runtimes and they are not applicable at an early stage.

Figure 11.6 (right) shows nearly perfect quality for EDCAR. The quality of the other methods decreases to different extent. Many object sets are regarded as potential clusters for this setting. Thus, with a simple redundancy model as used in CoPaM and Cocain° also uninteresting clusters are selected for the result and

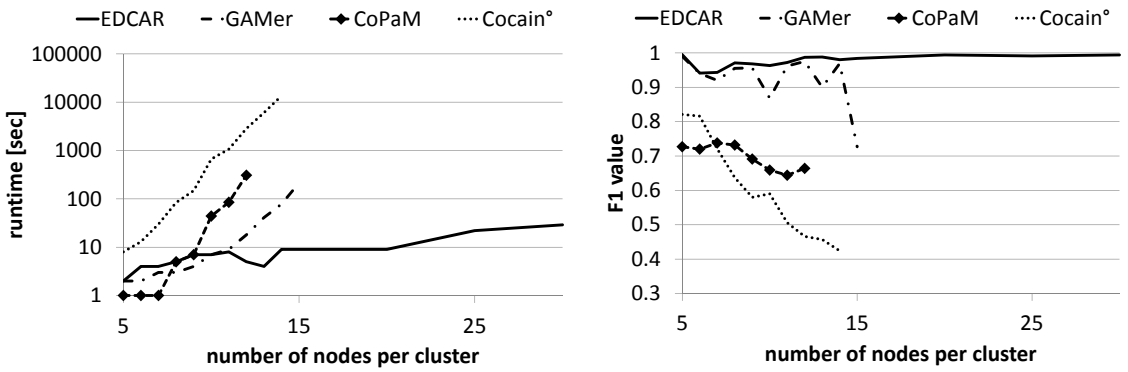


Figure 11.6: Scalability and quality w.r.t. cluster size

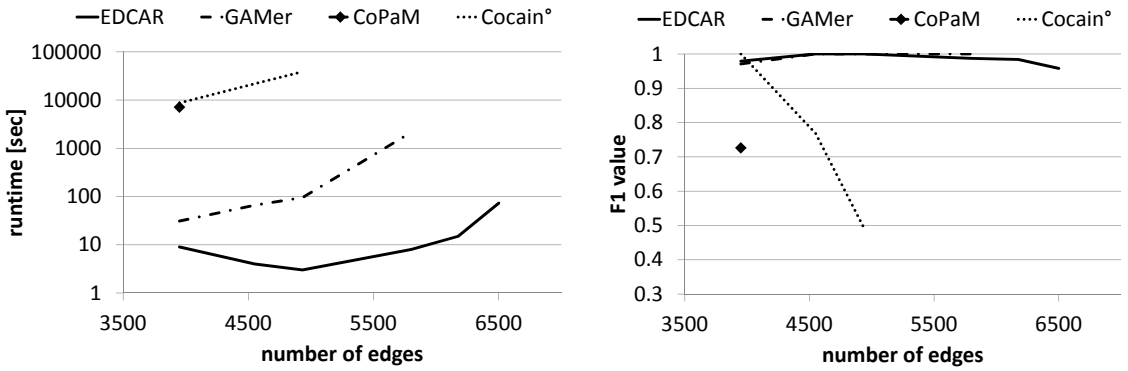


Figure 11.7: Scalability and quality w.r.t. cluster density

hence the quality is low. GAMER shows a slightly worse quality than EDCAR. The advantage of our novel clustering model becomes apparent.

Scalability w.r.t. graph density. In Figure 11.7 we increase the graph's density, i.e. we add edges between the clustered nodes. Accordingly, by keeping the minimal density threshold fixed, more subsets fulfill the cluster definition and the runtimes of the methods increase. Again, EDCAR is orders of magnitudes faster confirming the usefulness of our approximation techniques developed in this work. Please note that CoPaM was only able to determine the clustering for just a single dataset (first symbol in the plot).

Scalability w.r.t. cluster dimensionality. The effects of varying dimensionality of the hidden clusters are shown in Figure 11.8. For subspace clustering this property is challenging since exponential many subspace projections of each subspace cluster may exist. As illustrated, all algorithms are nearly not influenced by this characteristic; except for CoPaM, which again can only be applied for the smallest

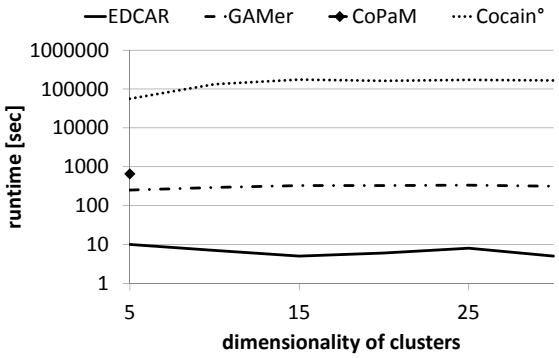


Figure 11.8: Scalability w.r.t. the dimensionality of the clusters

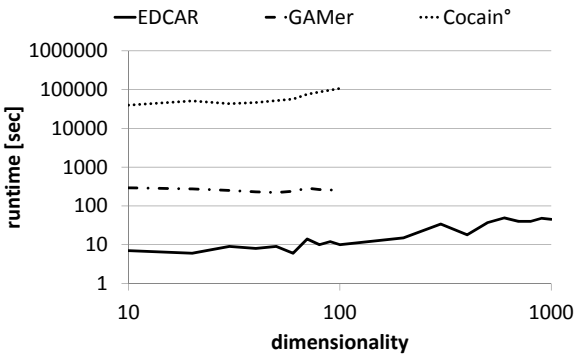


Figure 11.9: Scalability w.r.t. the dimensionality of the dataset

dataset. This experiment indicates that EDCAR is the most efficient method since its absolute runtime is once more the lowest. By traversing only small parts of the set enumeration tree, which is based on the object sets, EDCAR scales very well w.r.t. the cluster dimensionality.

Scalability w.r.t. data dimensionality. In Figure 11.9 we increase the data's dimensionality instead of the clusters' dimensionality. Though the runtimes of the competing methods do not increase significantly, they are not applicable for larger datasets due to the extreme memory usage. CoPaM is not applicable at all. The other methods have to manage a tremendous amount of clusters whereas our algorithm generates incrementally a small set of clusters.

Overall, all experiments indicate that EDCAR achieves far better runtimes than all competitors. EDCAR is the only method applicable to large datasets. At the same time the results of our approximation achieve high effectiveness.

Real world data. For the real world data in Figure 11.10 we cannot determine clustering qualities since no hidden clusters are given. Besides the runtime we

	Gene			Arxiv		Patent		DBLP
	EDCAR	GAMer	CoPam	EDCAR	GAMer	EDCAR	GAMer	EDCAR
runtime [s]	49	76	33060	179	945	282	574	25752
∅ nodes	9	8.8	9.67	7.7	8.2	12	11.7	9.3
∅ density	0.74	0.72	0.24	0.69	0.6	0.6	0.6	0.83
∅ dim.	15.8	15.47	12.21	5	5	3.0	3.0	3.02

Figure 11.10: Clustering properties for real world data

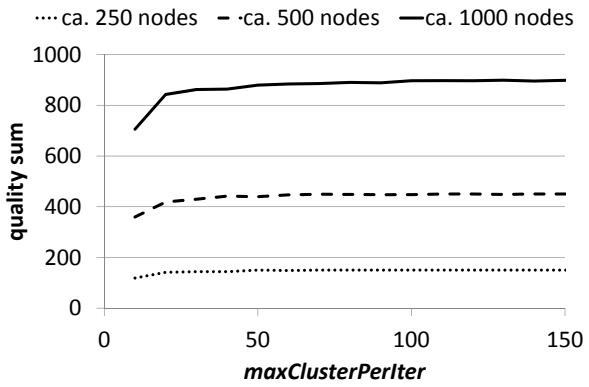


Figure 11.11: Generated paths per iteration of EDCAR

thus analyze different properties of the clustering results. For all datasets EDCAR is orders of magnitude faster than the other methods. GAMER is only applicable on three of the datasets. CoPaM can only be executed on the gene data and achieves extremely high runtimes. Cocain° finished on none of the datasets within 2 days. Considering the clustering result, we see that the clusters identified by EDCAR and GAMER have nearly similar properties; their characteristics differ only marginally. Thus, our approximations do not impair the clustering quality and EDCAR is able to detect interesting clusters in the data. Overall, EDCAR is the only method of choice for analyzing real world data.

Robustness. At last, we analyze the robustness of EDCAR w.r.t. the parameter $maxClusterPerIter$, which determines the number of selected paths and thus the maximal number of clusters generated in each iteration. By generating a large set, we can select the final clustering based on a larger portfolio of clusters; higher qualities should be obtained. However, large parameter values are accompanied by larger runtimes since more clusters are generated. In Figure 11.11 we vary this parameter for graphs with different sizes and we depict on the y-axis the overall quality sum of the final clustering (i.e. the criterion to be maximized). Very small parameter settings result in slightly smaller quality sums: too few and potentially not the most-interesting clusters are available for the final cluster selection. Using slightly higher settings, however, our method quickly converges towards the maximal quality sum for each graph. Even if we generate just a small set of clusters, we will get high qualities. EDCAR is robust w.r.t. the number of generated paths.

11.6 Conclusion

In this chapter we introduced an integrated mining method for efficiently detecting clusters showing high density in graphs as well as feature similarity in subspace projections. Our model combines subspace clustering and dense subgraph mining and performs an overall optimization of the result to get the most interesting, redundancy-free clustering. Based on the proven complexity of our model, we developed the algorithm EDCAR to efficiently calculate an approximate solution. By interweaving the process of cluster generation and cluster selection, which both make use of the GRASP principle, we determine high quality clusters and ensure low runtimes. Thorough experiments demonstrate that EDCAR has high effectiveness and at the same time constantly outperforms all competing approaches in terms of efficiency.

While both GAMER and EDCAR make use of the twofold cluster definition, we propose in the next chapter a more sophisticated cluster definition. This novel model is the first approach that exploits the advantages of density-based clustering in the domain of graph and attribute data.

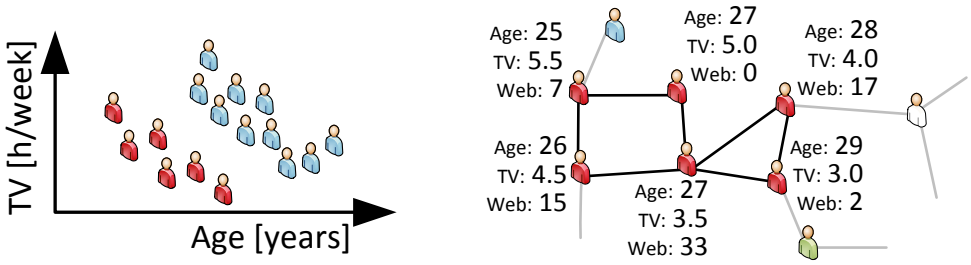
Chapter 12

Density-Based Subspace Clustering for Graph Data

In the previous chapters, we introduced integrated subspace clustering approaches successfully circumventing the problems of full-space clustering in heterogeneous attribute and graph data. For these initial solutions, we selected a simple cluster definition, which is limited to clusters of certain shapes. In this chapter we introduce a density-based cluster definition taking the attribute similarity in subspaces and the graph density into account. This novel cluster model enables us to detect more complex clusters of, e.g., arbitrary shape and size. Based on the novel notion of local densities, our DB-CSC model uses a fixed point iteration to find the desired clusters. Further pruning techniques, allow an efficient calculation of the overall clustering solution. In thorough experiments we demonstrate the strength of DB-CSC in comparison to related approaches.

12.1 Motivation and Comparison with Related Work

By our integrated subspace clustering methods introduced in the previous two chapters, we provide solutions to cluster objects represented by attribute data as well as network data. Since usually for each object a multitude of different characteristics is recorded – though not all of these characteristics are relevant for each cluster –, our methods perform dense subgraph mining in combination with subspace clustering, i.e. clusters are identified in subspace projections of the attribute data. An example is illustrated in Figure 12.1(b), where the highlighted



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

Figure 12.1: Clusters in a social network with attributes 'age', 'tv consume' and 'web consume'

group of customers is densely connected in the network as well as similar in two of their three attributes.

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

Though, the cluster models of GAMER (Chp. 10), EDCAR (Chp. 11) and CoPaM [MCRE09] are limited to clusters of certain shapes. Similar to grid-based subspace clustering [AGGR98], a cluster (w.r.t. the attributes) is simply defined by taking all objects located within a given grid cell, i.e. whose attribute values differ by at most a given threshold. The methods are biased towards small clusters with little extend. This drawback is even worsened by considering the used notions of dense subgraphs: e.g. by using quasi-cliques the diameter is a priori constrained to a fixed threshold [PJZ05]. Very similar objects just slightly located next to a cluster are lost due to such restrictive models. For real world data, such cluster definitions are usually too restrictive since clusters can exhibit more complex shapes. Considering for example the 2d attribute subspace of the objects in Figure 12.1(a): The two clusters can not be correctly detected by the rectangular hypercube model of CoPaM, GAMER, or EDCAR. Either both clusters are merged or some objects are lost. In Figure 12.1(b) an extract of the corresponding network structure is shown.

Since the restrictive quasi-clique property would only assign a low density to this cluster, the cluster would probably be split by the existing methods.

In this chapter, we combine dense subgraph mining with subspace clustering based on a more sophisticated cluster definition; thus solving the drawbacks of previous approaches. Established for other data types, density-based clustering techniques [EKSX96, SEKX98, HK98, KKK04, XYFS07, AKMS08a] have shown their strength in many scenarios. They do not require the number of clusters as an input parameter and are able to find arbitrarily shaped clusters. However, none of the introduced density-based clustering models considers graph data in combination with attribute data.

Even though the method of [XYFS07] analysis graph data, it solely transforms the network information to a (shared nearest neighbor) distance function. Similar the the method of [HZZL02], any distance-based clustering algorithm can be applied afterwards; in [XYFS07], the DBSCAN method is used. Thus, the work of [XYFS07] can simply be regarded as an instance of the generalized DBSCAN [SEKX98] principle. As already mentioned in Chapter 10.2, transforming the network to a distance function leads to results that are difficult to interpret as no conclusions about cluster structures in the graph are possible. Furthermore, [XYFS07] neither considers attribute data annotated to the vertices nor subspace clusters.

Thus, we introduce the first approach exploiting a *density-based clustering principle* to join the paradigms of subspace clustering and dense subgraph mining. Our clusters correspond to dense regions in the attribute space as well as in the graph. Based on local neighborhoods taking the attribute similarity in subspaces as well as the graph information into account, we model the density of single objects. By merging all objects located in the same dense region, the overall clusters are obtained. Thus, our model is able to detect the clusters in Figure 12.1 correctly.

Besides the sound definition of clusters based on density values, we achieve a further advantage. In contrast to previous approaches, the clusters in our model are not limited in their size or shape but can show arbitrary shapes. For example, the diameter of clusters based on quasi-cliques is a priori restricted by a parameter-dependent value [PJZ05] leading to a bias towards clusters of small size and little extent. Such a bias is avoided in our model, where the sizes and shapes of clusters are automatically detected. Overall, our contributions are:

- We introduce a density-based cluster definition taking attributes in subspaces and graph information into account.

- We ensure an unbiased cluster detection since our clusters can have arbitrary shape and size due to our novel concept of local densities.
- We show that our clusters can be calculated using a fixed point iteration and we develop the algorithm DB-CSC to determine our clustering solution.

12.2 A Density-Based Clustering Model for Combined Data

In this section we introduce our density-based clustering model for the combined clustering of graph data and attribute data. The clusters in our model correspond to dense regions in the graph as well as in the attribute space. For simplicity, we first introduce in Section 12.2.1 the cluster model for the case that only a single subspace, e.g. the full-space, is considered. The extension to subspace clustering and the definition of a redundancy model to confine the final clustering to the most interesting subspace clusters is introduced in Section 12.2.2, while a discussion of this model is presented in Section 12.2.3.

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

12.2.1 Cluster model for a single subspace

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

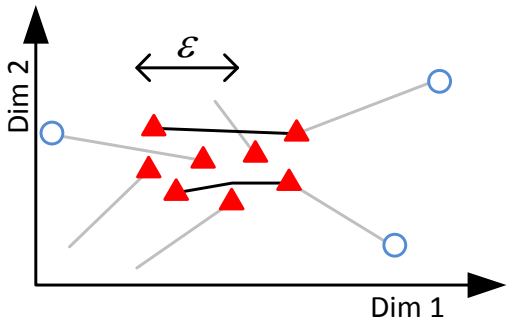


Figure 12.2: Dense region in the attribute space but sparse region in the graph

If we would simply determine the neighborhood based on the attribute data, as done in [EKSX96, HK98], we could define the attribute neighborhood of a vertex v by its ε-*neighborhood*, i.e. the set of all objects which distances to o do not exceed a threshold ε. Formally,

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

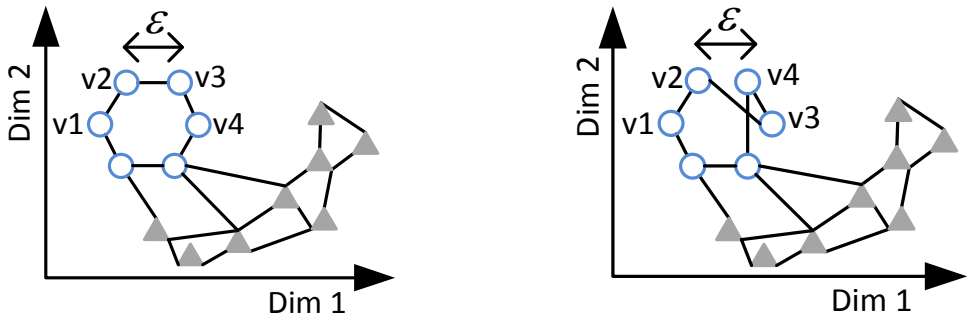
with an appropriate distance function as, e.g., the maximum norm $dist(x, y) = \max_{i \in \{1,\ldots,D\}} |x[i] - y[i]|$. Just considering attribute data, however, leads to the problem illustrated in Figure 12.2. Considering the attribute space, the red triangles form a dense region. (The edges of the graph and the parameter ε are depicted in the figure.) Though, this group is not a meaningful cluster since in the graph this vertex set is not densely connected. Accordingly, we have to consider the graph data and attribute data simultaneously to determine the neighborhood.

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

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

In Figure 12.2 the red triangles would not be dense anymore because their simple combined neighborhoods are empty. However, using just the adjacent vertices leads to a too restrictive cluster model, as the next example in Figure 12.3 shows.

Assuming that each vertex has to contain three objects in its neighborhood (including the object itself) to be dense, we get two densely connected sets, i.e. two clusters, in Figure 12.3(a). In Figure 12.3(b), we have the same vertex set, the



(a) Successful with $k \geq \mathbf{1}$, $minPts = 3$ (b) Successful with $k \geq \mathbf{2}$, $minPts = 3$

Figure 12.3: Robust cluster detection by using k-neighborhoods. Adjacent vertices ($k = 1$) are not always sufficient for correct detection (left: successful; right: fails)

same set of attribute vectors and the same graph density. The example only differs from the first one by the interchange of the attribute values of the vertices v_3 and v_4, which both belong to the same cluster in the first example. Intuitively, this set of vertices should also be a valid cluster. However, it is not because the neighborhood of v_2 contains just the vertices $\{v_1, v_2\}$. The vertex v_4 is not considered since it is only similar to v_2 w.r.t. the attributes but not adjacent to v_2.

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

Definition 12.1 (Graph k-neighborhood) *As vertex u is k-reachable from a vertex v (over a set of vertices V) if*

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

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

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

Please note that the object v itself is contained in its neighborhood $N_k^V(v)$ as well. Overall, the combined neighborhood of a vertex $v \in V$ considering graph

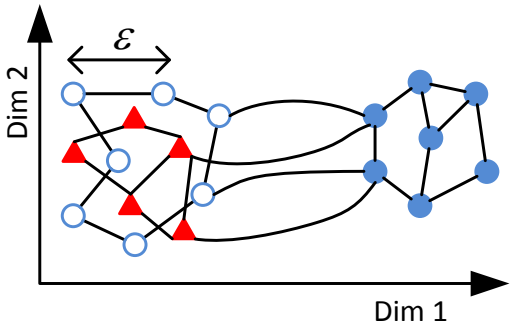


Figure 12.4: Novel challenge by using local density computation and the graph k-neighborhood

and attribute data can be formalized by intersecting v's graph k-neighborhood with its ε-neighborhood.

Definition 12.2 (Combined local neighborhood) *The combined neighborhood of* $v \in V$ *is:*

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

Using the combined neighborhood $N^V(v)$ and $k \geq 2$, we get in Figure 12.3(a) and Figure 12.3(b) the same two clusters. In both examples v_2's neighborhood contains 3 vertices, e.g. $N^V(v_2) = \{v_1, v_2, v_4\}$ in Figure 12.3(b). So to speak, we "jump over" the vertex v_3 to find further vertices that are similar to v_2 in the attribute space.

Local density calculation. As mentioned, the "jumping" principle is necessary to get meaningful clusters. However, it leads to a novel challenge not given in previous density-based clustering approaches. Considering Figure 12.4, the vertices on the right hand side form a combined cluster and are clearly separated from the remaining vertices by their attribute values. However, on the left hand side we have two separate clusters, one consisting of the vertices depicted as dots and one consisting of the vertices depicted as triangles. If we would only consider attribute data for clustering, these two clusters would be merged into one big cluster as the vertices are very similar w.r.t. their attribute values. If we would just use adjacent vertices, this big cluster would not be valid as there are no edges between the dot vertices and the triangle vertices. However, since we allow "jumps" over vertices, the two clusters could be merged, e.g. for $k = 2$.

This problem arises because the dots and triangles are connected via the vertices on the right hand side, i.e. we "jump" over vertices that actually do not belong to the final cluster. Thus, a "jump" has to be restricted to the objects of the same cluster. In Figure 12.3(b) e.g., the vertices $\{v_2, v_3, v_4\}$ belong to the same cluster. Thus, reaching v_4 over v_3 to increase the density of v_2 is meaningful. Formally, we have to restrict the k-reachability used in Definition 12.1 to the vertices contained in the cluster O. In this case, the vertices on the right hand side of Figure 12.4 cannot be used for "jumping" and thus the dots are not in the triangles' neighborhoods and vice versa. Thus, we are able to separate the two clusters.

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

In our theoretical clustering model, we can solve this cyclic dependency by assuming a set of clustered objects O as given. The algorithmic solution is presented in Section 12.3. Formally, given the set of clustered objects O three properties have to be fulfilled: First, each vertex $v \in O$ has to be dense w.r.t. it local neighborhood. Second, the spanned region of these objects has to be (locally) connected since otherwise we would have more than two clusters (cf. Figure 12.4). Last, the set O has to be maximal w.r.t. the previous properties since otherwise some vertices of the cluster are lost. Overall, we define:

Definition 12.3 (Density-based combined cluster)
A combined cluster in a graph $G = (V, E, l)$ w.r.t. the parameters k, ε and $minPts$ is a set of vertices $O \subseteq V$ that fulfills the following properties:

(1) high local density: $\forall v \in O : |N^O(v)| \geq minPts$

(2) locally connected: $\forall u, v \in O : \exists w_1, \ldots, w_l \in O : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : w_i \in N^O(w_{i+1})$

(3) maximality: $\neg\exists O' \supset O : O'$ *fulfills (1) and (2)*

Please note that the neighborhood calculation $N^O(v)$ is always done w.r.t. to the set O and not w.r.t. the whole graph V. Based on this definition and by using $minPts = 3$, $k \geq 2$, we can e.g. detect the three clusters from Figure 12.4. By using a too small value for k (as for example $k = 1$ in Figure 12.3(b)), clusters are often split or even not detected at all. On the other hand, if we choose k too high, we run the risk that the graph structure is not adequately considered any more. However, for the case depicted in Figure 12.4 our model detects the correct clusters even for arbitrary large k values as the cluster on the right hand side is clearly separated from the other clusters by its attribute values. The two clusters on the left hand side are never merged by our model as we do not allow jumps over vertices outside the cluster.

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

12.2.2 Overall subspace clustering model

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

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

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

Definition 12.4 (Density-based combined subspace cluster)
A combined subspace cluster $C = (O, S)$ in a graph $G = (V, E, l)$ consists of a set of vertices $O \subseteq V$ and a set of relevant dimensions $S \subseteq Dim$ such that O forms a combined cluster (cf. Def 12.3) w.r.t. the local subspace neighborhood $N^O_S(v) = N^O_k(v) \cap N^O_{\varepsilon,S}(v)$ with $N^O_{\varepsilon,S}(v) = \{u \in O \mid dist^S(l(u), l(v)) \leq \varepsilon\}$.

As we will show, our subspace clusters fulfill an anti-monotonicity property, which we will exploit in our algorithm to find the valid clusters more efficiently.

Theorem 12.1 (Anti-monotonicity) *For every subspace cluster $C = (O, S)$ and every $S' \subseteq S$ there exists a vertex set $O' \supseteq O$ such that (O', S') is again a valid cluster.*

Proof 12.1 *For every two subspaces S, S' with $S' \subseteq S$ and every pair of vertices u, v it holds that $dist^{S'}(l(u), l(v)) \leq dist^{S}(l(u), l(v))$. Thus for every vertex $v \in O$ we get $N^O_{S'}(v) \supseteq N^O_S(v)$. Accordingly, the properties (1) and (2) from Def. 12.3 are fulfilled by (O, S'). If (O, S') is maximal w.r.t. these properties, then (O, S') is a valid combined subspace cluster. Else, by definition there exists a vertex set $O' \supset O$ such that (O', S') is a valid subspace cluster.* □

Redundancy removal. Because of the density-based subspace cluster model, there can not be an overlap between clusters in the same subspace. However, a vertex can belong to several subspace clusters in different subspaces, thus clusters from different subspaces can overlap. Due to the exponential number of possible subspace projections, an overwhelming number of (very similar) subspace clusters may exist. Whereas allowing overlapping clusters in general makes sense in many applications, allowing too much overlap can lead to highly redundant information. Thus, for meaningful interpretation of the result, removing redundant clusters is crucial. The redundancy removal in our novel method is similar to the ones introduced in our GAMER approach.

First, we define the interestingness of a cluster. The combined subspace clusters are characterized by the two important properties "number of vertices" and "dimensionality". Since both measures cannot be optimized simultaneously (cf. anti-monotonicity), we again trade them off in a single objective function. Please note that in contrast to our GAMER approach, we do not consider the density of the clusters. Since, first, the density of any combined subspace clusters is lower bounded by $minPts$ and second according to Definition 12.7 we select maximal object sets, it is very likely that at least one vertex of the cluster fulfills the equality $|N^O(v)| = minPts$ (and not the inequality $>$). Thus, with high probability the density of any cluster is around $minPts$ and, hence, a consideration of such a constant value in the objective function is not necessary. Overall, we define the interestingness of a single cluster as follows:

Definition 12.5 (Interestingness of a cluster)
The interestingness of a combined subspace cluster $C = (O, S)$ is computed as

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

For the selection of the final clustering based on this interestingness definition we use the same redundancy definition as introduced by GAMER. For completeness, we review the two important definitions. First, the redundancy between two clusters is given as follows:

Definition 12.6 (Redundancy between clusters) *Given the two redundancy parameters $r_{obj}, r_{dim} \in [0, 1]$, the binary redundancy relation $\prec_{red}$ is defined by:*

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

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

Definition 12.7 (Optimal density-based combined subspace clustering)
Given the set of all combined clusters $Clusters$, the optimal combined clustering $Result \subseteq Clusters$ fulfills

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

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

12.2.3 Discussion of Clustering Model and Parameters

Our cluster model can be adapted based on the three parameters ε, k, and $minPts$, whose effects we discuss in the following. We use the gained insights to develop an algorithm that resolves the cyclic dependency of our cluster definition in the next section.

Let us first consider the parameter ε: By increasing ε the neighborhood sizes regarding the attribute domain increase. For the extremal value of $\varepsilon \to \infty$, all objects are considered similar to each other, i.e. $N_\varepsilon^V(v) = V$. Or even more precise, we get $N_\varepsilon^O(v) = O$ for all $O \subseteq V$. In this case, the combined neighborhood $N^O(v)$ (cf. Definition 12.2) can be simplified to the graph k-neighborhood $N_k^O(v)$ and, thus, in our overall cluster definition we just consider the graph data. Accordingly, the results for each subspace S are identical and our method just has to analyze the full-dimensional space.

For this scenario and choosing $k = 1$, we can show that the clusters determined by our method correspond to the well-known notion of x-cores [DGM06, JL07].

Definition 12.8 (x-core) *An x-core O in a graph $G = (V, E, l)$ is a maximal connected subgraph $O \subseteq V$ in which all vertices connect to at least x other vertices in O, i.e. $\forall v \in O : deg_G(v, O) \geq x$ where $deg_G(v, O) = |\{u \in O \mid (v, u) \in E\}|$.*

Using this definition, we show the following theorem:

Theorem 12.2 (Relation between clusters and cores) *If $\varepsilon \to \infty$ and $k = 1$, each combined subspace cluster $C = (O, S)$ in the graph $G = (V, E, l)$ (cf. Definition 12.3 and 12.4) corresponds to a $(minPts-1)$-core in G.*

Proof 12.2 *Due to property (1) of Def. 12.3 we get: high local density $\Leftrightarrow \forall v \in O : |N^O(v)| \geq minPts \overset{k=1}{\Leftrightarrow} \forall v \in O : |N_k^O(v)| \geq minPts \overset{\varepsilon \to \infty}{\Leftrightarrow} \forall v \in O : |\{u \in O \mid (v, u) \in E\} \cup \{v\}| \geq minPts \Leftrightarrow \forall v \in O : deg_G(v, O) \geq minPts - 1$*
Due to property (2) of Def. 12.3 we get: locally connected $\overset{\varepsilon \to \infty, k=1}{\Leftrightarrow} \forall u, v \in O : \exists w_1, \ldots, w_l \in O : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : (w_i, w_{i+1}) \in E \Leftrightarrow O$ is connected in G
Due to property (3) of Def. 12.3, $C = (O, S)$ is a maximal subgraph fulfilling the above properties. Thus, $C = (O, S)$ is a $(minPts-1)$-core in G. □

This observation just holds for the simple case $k = 1$, but we will show in the next section how to extend this idea to solve the general setting with $k > 1$. Before this, however, we analyze the parameter k: Similar to ε, increasing k leads to larger neighborhood sizes; in this case w.r.t. the graph information. Assuming a connected graph and the extremal value of $k \to \infty$, all vertices are located in the graph neighborhood, i.e. $N_k^V(v) = V$. However, we *cannot* infer that $N_k^O(v) = O$ for all $O \subseteq V$. Due to the local density principle, we are only

allowed to consider the vertices of O and we (implicitly) remove the vertices $V \setminus O$, resulting in a potentially disconnected graph where we cannot reach all vertices, i.e. $N_k^O(v) \neq O$ might hold. As a consequence, for the scenario $k \to \infty$ it is *not* possible to simplify the combined neighborhood. Thus, even for $k \to \infty$ we exploit both sources: the graph data and the attribute data (in contrast to $\varepsilon \to \infty$, where we only consider the graph data).

A special case, in which we *can* simplify the combined neighborhood, is assuming a complete input graph. Obviously, in this case, the graph structure is not (and need not to be) considered by our model. Thus, any choice of k leads to the same result and we only detect clusters based on the attribute values. For this setting, our model is related to traditional density-based clustering methods such as DBSCAN [EKSX96] or its extension to subspace clustering SUBCLU [KKK04]. However, our model differs in two aspects: First, our model removes redundant clusters to prevent an overwhelming result size. Second, the methods of [EKSX96, KKK04] include so called border objects. These objects correspond to objects whose densities do not exceed the value of $minPts$ but are located in the neighborhood of a dense object. To include border objects in our model, we can use a tuple (O, O_b) to distinguish between the different types of objects in a cluster and we have to slightly modify Definition 12.3 to

(1) high local density or border object: $\forall v \in O : |N^{O \cup O_b}(v)| \geq minPts$ and $\forall v \in O_b : \exists u \in O : v \in N^{O \cup O_b}(u)$

(2) locally connected: as before

(3) maximality: $\neg\exists (O', O_b')$ with $(O' \cup O_b') \supset (O \cup O_b)$ and (O', O_b') fulfills (1) & (2)

Please note that Property (2) remains unaffected: one has to ensure the local connectivity of O, i.e. one cannot use border objects to connect the vertices from O. Otherwise, an object with low density could potentially merge two different dense regions into a single clusters (similar to Figure 12.4). As a side effect of this modification, the resulting clusters in a single subspace are not necessarily disjoint any more. Instead, a border object can belong to more than one cluster as it does not merge the two clusters.

At last, we discuss the parameter $minPts$. While increasing the values of k or ε leads to larger clusters due to larger neighborhood sizes, the parameter $minPts$ causes the inverse effect: the larger $minPts$, the smaller the clusters. The minimal density threshold is harder to reach and fulfilled by less objects. Consequently, the choice $minPts \to \infty$ is not meaningful. For the case $minPts \leq 2$ (and again

$k = 1$) our model returns for each subspace S the connected components of the graph G'_S that results by removing every edge (of the original graph) connecting dissimilar nodes. Formally,

Theorem 12.3 (Relation between clusters and connected components)
If $minPts \leq 2$ and $k = 1$, any combined subspace cluster $C = (O, S)$ in the graph $G = (V, E, l)$ (cf. Definition 12.3 and 12.4) corresponds to a connected component of the graph G'_S=(V, E'_S, l) with $E'_S = \{(v, u) \in E \mid dist^S(l(v), l(u) \leq \varepsilon)\}$.

Proof 12.3 *Due to property (2) of Def. 12.3 we get: locally connected $\overset{k=1}{\Leftrightarrow} \forall u, v \in O: \exists w_1, \ldots, w_l \in O : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : (w_i, w_{i+1}) \in E'_S \Leftrightarrow O$ is connected in G'_S.*
Since O is connected, each $v \in O$ fulfills $|N^O(v)| \geq 2 \geq minPts$. Thus, property (1) of Def. 12.3 is directly fulfilled.
Due to property (3) of Def. 12.3, $C = (O, S)$ is a maximal connected subgraph of G'_S. That is, C is one of G'_S's connected components. □

Note that Theorem 12.3 holds for arbitrary ε values, while Theorem 12.2 requires $\varepsilon \to \infty$ but allows arbitrary $minPts$ values. Combining both theorems, i.e. $\varepsilon \to \infty$ and $minPts \leq 2$, we have shown as a nice byproduct the well-known observation that any 1-core in graph G corresponds to one of its connected components. In the next section we utilize the notion of cores to solve our general clustering model for arbitrary parameter settings and especially for the complex case of $k \geq 2$.

12.3 The DB-CSC Algorithm

In the this section we describe the DB-CSC (**D**ensity-**B**ased **C**ombined **S**ubspace **C**lustering) algorithm for detecting the optimal clustering result. In Section 12.3.1 we present the detection of our density-based clusters in a single subspace S and in Section 12.3.2 we introduce the overall processing scheme using different pruning techniques to enhance the efficiency.

12.3.1 Finding clusters in a single subspace

Before we present our method to detect the combined clusters in subspace S, we first introduce some basic definitions. Theoretically, each subset of objects $O \subseteq V$

could be evaluated whether it forms a valid cluster in the subspace S, and selecting some of these sets results in a potential clustering $\mathcal{C}$. We denote with A the set of all possible groups in subspace S, i.e. $A = \{(O, S) \mid O \subseteq V\}$, and with $\mathcal{A}$ the set of all possible groupings in subspace S, i.e. $\mathcal{A} = \mathcal{P}(A)$. Obviously, one of these groupings $\mathcal{R} \in \mathcal{A}$ has to correspond to the (so far unknown) clustering solution in the subspace S. Given the set $\mathcal{A}$, we first define the following binary relation to compare different groupings:

Definition 12.9 (Strict partial order of groupings) *Let $A = \{(O, S) \mid O \subseteq V\}$ be the set of all possible groups in subspace S and $\mathcal{A} = \mathcal{P}(A)$ be the set of all possible groupings in subspace S. The strict partial order $<_S^{\mathcal{A}}$ over the set of all possible groupings $\mathcal{A}$ in subspace S is defined as*

$\forall \mathcal{C}_1, \mathcal{C}_2 \in \mathcal{A} : \mathcal{C}_1 <_S^{\mathcal{A}} \mathcal{C}_2 \Leftrightarrow (Cov(\mathcal{C}_1) \subset Cov(\mathcal{C}_2)) \vee (Cov(\mathcal{C}_1) = Cov(\mathcal{C}_2) \wedge |\mathcal{C}_1| > |\mathcal{C}_2|)$
with $Cov(\mathcal{C}) = \bigcup_{(O_i,S)=C_i \in \mathcal{C}} O_i$

Intuitively, the grouping $\mathcal{C}_2$ is "better" than the grouping $\mathcal{C}_1$ ($\mathcal{C}_1 <_S^{\mathcal{A}} \mathcal{C}_2$), if the object groups contained in $\mathcal{C}_2$ cover a larger amount of objects or if the coverage is identical but $\mathcal{C}_2$ needs less sets to obtain such a coverage. In the following, we use this partial order to prove our main theorem for detecting the clustering solution.

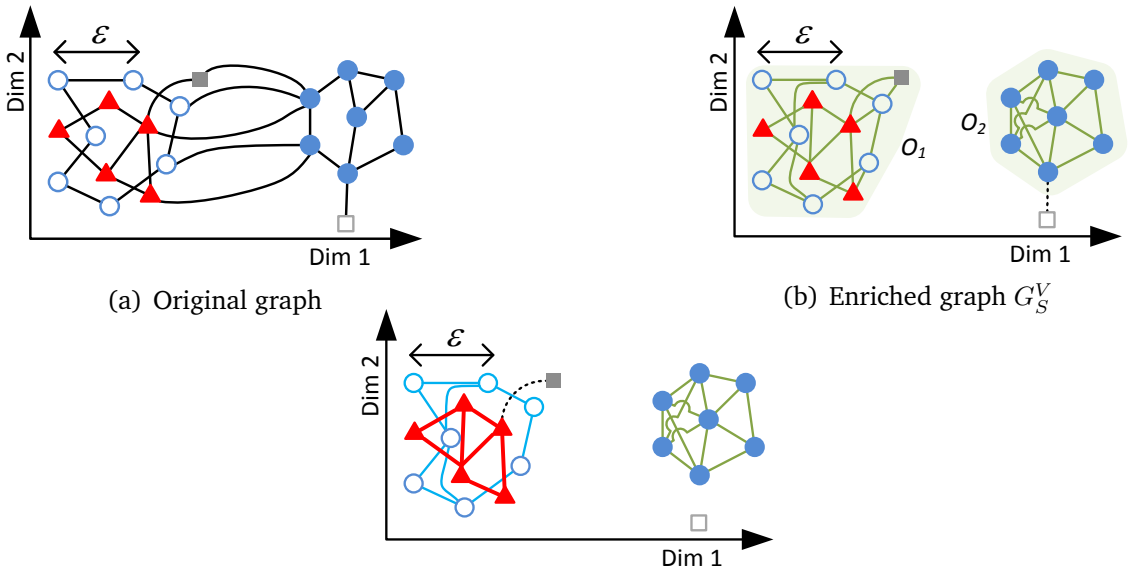
Clustering by Fixed Point Iteration

Given the set of all possible groupings $\mathcal{A}$, we are interested in finding the correct element $\mathcal{R} \in \mathcal{A}$ that corresponds to the clustering solution in subspace S, i.e. we want to detect all but only the object sets $O \subseteq V$ fulfilling Definition 12.3/12.4. To achieve this aim, we transfer the observations of Theorem 12.2 to the general case of arbitrary ε values and $k > 1$.

To make Theorem 12.2 applicable, we first introduce a graph transformation that represents attribute and graph information simultaneously.

Definition 12.10 (Enriched subgraph) *Given a set of vertices $O \subseteq V$, a subspace S, and the original graph $G = (V, E, l)$, the enriched subgraph $G_S^O = (V', E')$ is defined by $V' = O$ and $E' = \{(u, v) \mid v \in N_S^O(u) \wedge v \neq u\}$ using the distance function $dist^S$.*

Please note that the enriched subgraph is calculated based on a given vertex set O, which is not necessarily the whole vertex set V. Two vertices are adjacent in the enriched subgraph iff their attribute values are similar in S and the vertices are



(a) Original graph

(b) Enriched graph G_S^V

(c) $G_S^{O_i}$ for subsets O_i

Figure 12.5: Finding clusters by $(minPts-1)$-cores in the enriched graphs (example with k=2, $minPts$=3)

connected by at most k edges in the original graph (using just vertices from O). In Fig. 12.5(b) the enriched subgraph for the whole set of vertices V is computed, while Fig. 12.5(c) just considers the subset O_1 and O_2 respectively.

Since in the enriched subgraph the neighborhood information based on the attribute domain is directly encoded in the graph structure (using the original value of ε), we can select in the transformed graph $\varepsilon \to \infty$. Similarly, the (original) graph k-neighborhood is encoded by adjacent vertices in the enriched subgraph; we can select $k = 1$. Thus, Theorem 12.2 is applicable on the enriched subgraph, i.e. in this graph we can concentrate on the detection of $(minPts-1)$-cores for our cluster detection. However, the detected cores do not exactly represent our clusters!

While constructing the enriched subgraph G_S^O, we consider the neighborhoods based on all vertices of O. However, due to our local density calculation, we are only allowed to use the clustered objects O' for the neighborhood calculation, which are potentially a subset of the objects O. Thus, as illustrated in Figure 12.5(b), the detected cores can still merge several clusters. Though, what is true is the following: Each cluster O' is a subset of some $(minPts-1)$-core in G_S^V. More precisely, we can prove this property for each enriched subgraph G_S^O with $O \supseteq O'$.

Theorem 12.4 (Subset property w.r.t. cores) *Let $C = (O', S)$ be a combined subspace cluster, $O \supseteq O'$, and G_S^O the enriched subgraph w.r.t. O. There exists a $(minPts-1)$-core $X \subseteq O$ in G_S^O with $O' \subseteq X$.*

Proof 12.4 *a) Since $O \supseteq O'$, we get $N_S^O(u) \supseteq N_S^{O'}(u)$ for all $u \in V$ and thus G_S^O contains all edges of $G_S^{O'}$.*
b) Due to property (1) of Def. 12.3 for the cluster C it holds: high local density $\Leftrightarrow$ $\forall v \in O' : |N^{O'}(v)| \geq minPts \Leftrightarrow \forall v \in O' : |N^{O'}(v) \backslash \{v\}| \geq minPts - 1 \Leftrightarrow \forall v \in O' : deg_{G_S^{O'}}(v, O') \geq minPts - 1 \overset{a)}{\Rightarrow} \forall v \in O' : deg_{G_S^O}(v, O') \geq minPts - 1 \Rightarrow$ exists maximal set $X \supseteq O'$ with $\forall v \in X : deg_{G_S^O}(v, X) \geq minPts - 1$
c) Due to property (2) of Def. 12.3 for the cluster C it holds: locally connected $\Leftrightarrow$ $\forall u, v \in O' : \exists w_1, \ldots, w_l \in O' : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : w_i \in N_S^{O'}(w_{i+1})$ $\Leftrightarrow \forall u, v \in O' : \exists w_1, \ldots, w_l \in O' : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : (w_i, w_{i+1})$ are adjacent in $G_S^{O'} \Leftrightarrow G_S^{O'}$ is connected $\overset{a)}{\Rightarrow} O$ belongs to a single connected component of G_S^O
d) Based on b+c: there exists a connected maximal set $X \supseteq O'$ with $\forall v \in X : deg_{G_S^O}(v, X) \geq minPts - 1 \Rightarrow X$ is a $(minPts-1)$-core in G_S^O with $O' \subseteq X$ □

Theorem 12.4 implies that our algorithm only has to analyze vertex sets that potentially lead to $(minPts-1)$-cores. If we were able to guess the cluster O', then the detected core in $G_S^{O'}$ would exactly correspond to our cluster since in this case $O{=}O'$ and thus $O' \subseteq X \subseteq O'$. Based on these observations, we show:

Theorem 12.5 (Fixed Point Solution) *Let $<_S^{\mathcal{A}}$ be the strict partial order over the set of all possible groupings $\mathcal{A}$ in subspace S.*
1) The set of all combined clusters in subspace S corresponds to the largest fixed point of the function $f_S : \mathcal{A} \to \mathcal{A}$ with

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

2) The sequence $f_S^\infty(\{(V, S)\})$ converges to this fixed point

Proof 12.5 *Let $\mathcal{R} \in \mathcal{A}$ be the set of all combined clusters in subspace S.*
a) $\mathcal{R}$ is a fixed point of f_S. Due to Theorem 12.4 there exists for each cluster $C_i \in \mathcal{R}$ a $(minPts-1)$-core $X \subseteq O_i$ in $G_S^{O_i}$ with $O_i \subseteq X$. Thus, $X = O_i$ has to hold. Since all vertices of $G_S^{O_i}$ are covered by X there cannot exist further cores in $G_S^{O_i}$. Thus, $f_S(\{C_i\}) = \{C_i\}$ and overall $f_S(\mathcal{R}) = \mathcal{R}$.

b) $\mathcal{R}$ is the largest fixed point. Assume $\mathcal{R}$ is not the largest fixed point, i.e. there exists $\mathcal{C} \in \mathcal{A}$ with $f_S(\mathcal{C}) = \mathcal{C}$ and $\mathcal{R} <_S^{\mathcal{A}} \mathcal{C}$. Thus, due to Definition 12.9 either $Cov(\mathcal{C}) \supset Cov(\mathcal{R})$ or $Cov(\mathcal{C}) = Cov(\mathcal{R})$ with $|\mathcal{C}| < |\mathcal{R}|$ has to hold. The first case cannot hold since $\mathcal{R}$ contains all maximal clusters (Prop. (3), Def. 12.3), i.e. the coverage of $\mathcal{R}$ is maximal, and thus $Cov(\mathcal{C}) \subseteq Cov(\mathcal{R})$. Thus, the second case with $|\mathcal{C}| < |\mathcal{R}|$ has to hold. To realize this, at least two clusters $C_i, C_j \in \mathcal{R}$ have to be (partially) merged into a single group $D \in \mathcal{C}$. In this case, however, D contains at least two ($minPts-1$)-cores, i.e. $|f_S(\{D\})| \geq 2$ and especially $f_S(\{D\}) \neq \{D\}$. Thus, $\{D\}$ and therefore $\mathcal{C}$ are not fixed points of f_S. This is a contradiction to the assumption. Hence, $\mathcal{R}$ is the largest fixed point.

c) It holds $f_S^\infty(\{(V,S)\}) = \mathcal{R}$. First, we do not miss (or split up) any cluster $C \in \mathcal{R}$ by applying the sequence f_S^∞. Due to Theorem 12.4 each cluster $(O',S) \in \mathcal{R}$ is contained in a ($minPts-1$)-core of the graph G_S^O with $O \supseteq O'$. Especially it has to be contained within a core of the graph G_S^V. Since in each iteration $f_S(\mathcal{C}_i) = \mathcal{C}_{i+1}$ the whole set of cores is returned, there exists due to Theorem 12.4 one core $(O,S) \in \mathcal{C}_{i+1}$ with $O \supseteq O'$. Furthermore, since cores are disjoint due to their maximality, the result of $f_S^\infty(\{(V,S)\})$ cannot contain subsets of clusters. Second, the result of $f_S^\infty(\{(V,S)\})$ only contains the clusters of $\mathcal{R}$. Assume a group $D \in f_S^\infty(\{(V,S)\})$ which is not a valid cluster (and not one of its subsets). Based on Statement 1 of Theorem 12.5 we know that $\{D\}$ cannot be a fixed point. Thus, there exists a k such that $D \notin f_S^m(\{(V,S)\})$ for all $m > k$. This is a contradiction to the assumption. Overall, $f_S^\infty(\{(V,S)\}) = \mathcal{R}$. □

Overall, Statement 2 of Theorem 12.5 provides us with a solution to our clustering problem. The clusters in a single subspace S can be detected by a fixed point iteration. Thus, we are able to resolve the cyclic dependency. Algorithmically, we start by extracting from G_S^V all ($minPts-1$)-cores, since only these sets could lead to valid clusters. In Fig. 12.5(b) these sets are highlighted. However, keep in mind that these ($minPts-1$)-cores do not necessarily correspond to valid clusters. Theorem 12.5 requires a fixed point. In Figure 12.5(b) for example, we will get two cores but, as already discussed, the left set O_1 is not a valid cluster but has to be split up. Thus, we recursively have to apply the function until the fixed point is reached. Intuitively, in each iteration the clusters are refined until the final clusters are obtained.

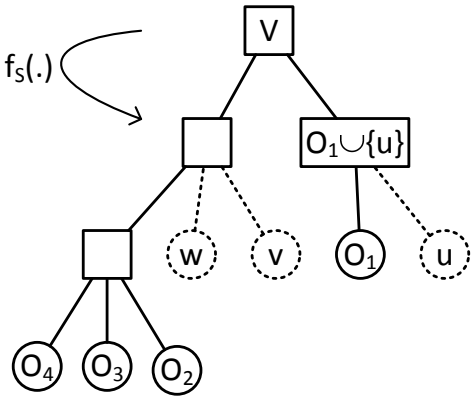


Figure 12.6: Illustration of fixed point iteration method

In Figure 12.6 we depict a schematic representation of the vertex sets to be analyzed in our method. Similar to a dendrogram, we start in the root with the whole set of vertices V. By applying the function f_S, smaller cores are detected. These cores are depicted as child nodes with solid lines in the dendrogramm. Vertices that do not belong to any core are depicted with dashed lines. By recursively applying f_S the cores are refined until the final clusters are obtained (leaf nodes with solid lines in the tree). Thus, in the example, four different clusters C_i with their corresponding object sets O_i are detected.

In summary, detecting density based clusters in combined data sources is far more complex than in traditional vector data. While existing clustering approaches can simply determine the density values for each object independently [EKSX96, KKK04], in our method such an independent computation is not possible due to the local density calculation. Please note that the toy example in Figure 12.5(a) might lead to the idea that we can simply remove all edges connecting dissimilar nodes and then perform a simple clustering on the residual graph. This a-priori rejection of edges, however, is not possible. As discussed, the Figure 12.3(b) contains two clusters, which, however, will not be detected if we would remove the edge between v_2 and v_3. Indeed, this edge belongs to the cluster even if the two vertices are not directly similar. Overall, our fixed point iteration provides us with a solution to detect all combined clusters in the subspace S.

Convergence and Complexity

In this section we analyze the convergence and complexity of the fixed point iteration method. Important for practical applications is the convergence of the sequence $f_S^\infty(\{(V,S)\})$ after a finite number of iterations. It holds:

Theorem 12.6 (Convergence) *Let $\mathcal{R}$ be the set of all combined clusters in subspace S. The fixed point iteration started with $\{(V,S)\}$ converges after at most $|\mathcal{R}| + |V \backslash Cov(\mathcal{R})|$ iterations.*

Proof 12.6 *Using the representation of our method as illustrated in Fig. 12.6 or 12.7, this upper bound is easy to show. Obviously, the height of such a tree corresponds to the number of required iterations. For the worst case, we have to determine the maximal height among all possible trees (valid for the given clustering scenario): First, the number of leaf nodes in the tree is given by $|\mathcal{R}| + |V \backslash Cov(\mathcal{R})|$. Since each valid cluster corresponds to a solid leaf node in the tree representation, each inner node of the tree is the union of the object sets of its child nodes, and the root has to represent all objects V, we must have $|V \backslash Cov(\mathcal{R})|$ dashed leaf nodes. Second, the maximal height of a tree with $|\mathcal{R}|+|V \backslash Cov(\mathcal{R})|$ leaf nodes is achieved by a binary tree as illustrated in Fig. 12.7. This corresponds to the bound as given in Theorem 12.6.* □

To determine the overall complexity of the fixed point iteration we furthermore need to bound the size of the cores to be analyzed. Since an inner node O in the tree represents the union of all its child nodes, we can distinguish two cases: First, O contains objects that do not belong to any valid cluster, i.e. O is an inner node whose subtree contains leafs with dashed lines. In this case, $f_S(\{(O, S)\})$ must lead to cores that are subsets of O and for each $(O', S) \in f_S(\{C\})$ we have $|O'| < |O|$. Accordingly, the size of the largest child node of O in the tree is bounded by $|O| - 1$. Second, O contains only objects that belong to valid clusters. That is, the inner node O is exactly the union of several valid clusters $\{C_1, \ldots, C_n\} = \mathcal{R}' \subseteq \mathcal{R}$. In this case the largest core of $f_S(\{(O, S)\})$ can be bounded by $|O| - \min_{(O_i,S)\in\mathcal{R}'} |O_i|$ since $f_S(\{(O, S)\})$ cannot be a fixed point, the resulting cores must be supersets of the true clusters, and the cores need to be disjoint. Intuitively, this happens if the smallest cluster is split from O, while the remaining clusters are still merged into a single core. Based on these upper bounds for the largest core we can show the following theorem:

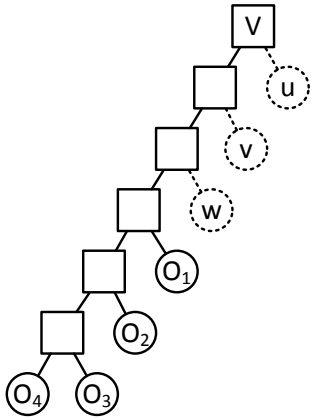


Figure 12.7: Worst case scenario

Theorem 12.7 (Complexity) *Given the graph $G = (V, E, l)$, the set $\mathcal{R}$ of all combined clusters in subspace S can be computed in time*

$$\mathcal{O}(|V \backslash Cov(\mathcal{R})| \cdot |V|^3 + |\mathcal{R}|^4 \cdot |O_{max}|^3)$$

with $(O_{max}, S) = \arg\max_{(O,S)\in\mathcal{R}}\{|O|\}$

Proof 12.7 *We first show that given a set of vertices $O \subseteq V$, the cores of the enriched subgraph G_S^O can be computed in cubic time: The graph G_S^O can be constructed by a restricted breadth-first search in the original graph, started at each node $v \in O$. Thus, in the worst case it requires time $\mathcal{O}(|O|^3)$. Detecting cores can be simply done by first recursively removing nodes with a node degree less than $minPts-1$ from G_S^O and then returning the connected components [DGM06, JL07]. The first step requires time $\mathcal{O}(|O|^2)$, since the removal of one node leads to an update of the degrees of the adjacent nodes. Determining the connected components also requires time $\mathcal{O}(|O|^2)$. Overall, the worst case runtime is dominated by constructing the enriched subgraph, i.e. time $\mathcal{O}(|O|^3)$.*

Now, we prove the overall complexity. First, as in the proof of Theorem 12.6, the worst case complexity occurs for a binary tree because each non-binary tree can be transformed to a binary one with larger runtime: W.l.o.g. we assume an inner node O with three child nodes O_1, O_2, O_3 is given. By replacing O's child nodes with O_1 and $X = O_2 \cup O_3$ as well as appending O_2, O_3 as child nodes to X, we get a binary tree representing the same clustering structure as before. Obviously, determining the clustering in this way leads to a higher runtime since the additional inner node X has to be analyzed.

Second, the worst case occurs if the cardinality of the object sets represented by inner nodes of the tree is as large as possible. Obviously, this happens if in the first $|V \backslash Cov(\mathcal{R})|$ iterations, only the non-clustered vertices are removed. That is, the first case of the above discussion applies and in iteration $i = 0, \ldots, |V \backslash Cov(\mathcal{R})| - 1$ we are just able to bound the size of the core to be analyzed by $|V| - i$. After these iterations, all non-clustered vertices are removed and the second case can be used: In the remaining $|\mathcal{R}|$ iterations, we split up in the worst case only the smallest of the clusters from the core. Let $x_1, \ldots, x_{|\mathcal{R}|}$ be the cluster sizes in increasing order, then we have to analyze in iteration j a core of at most size $\sum_{n=j}^{|\mathcal{R}|} x_n$.

Since each analysis step requires at most cubic time, we get

$$
\begin{aligned}
&\textstyle\sum_{i=0}^{|V \backslash Cov(\mathcal{R})|-1} (|V| - i)^3 + \sum_{j=1}^{|\mathcal{R}|} \left(\sum_{n=j}^{|\mathcal{R}|} x_n \right)^3 \\
&\leq \textstyle\sum_{i=0}^{|V \backslash Cov(\mathcal{R})|-1} |V|^3 + \sum_{j=1}^{|\mathcal{R}|} \left((|\mathcal{R}| - j + 1) \cdot x_{|\mathcal{R}|} \right)^3 \\
&\leq |V \backslash Cov(\mathcal{R})| \cdot |V|^3 + \textstyle\sum_{j=1}^{|\mathcal{R}|} (|\mathcal{R}|^3 \cdot |O_{max}|^3) \\
&\leq |V \backslash Cov(\mathcal{R})| \cdot |V|^3 + |\mathcal{R}|^4 \cdot |O_{max}|^3
\end{aligned}
$$

□

Please note that our worst case analysis assumes that in each iteration the enriched subgraph is computed from scratch, which results in the cubic complexity. In practice, however, we can extremely speed up the computation by exploiting the property that for each $O' \subseteq O$ the edges of $G_S^{O'}$ need to be a subset of the edges of G_S^O. Furthermore, we assume that for each subspace S we start with the whole vertex set V. In the following section, we show that the fixed point iteration can be initialized with far smaller object sets if we analyze multiple subspaces. In any case, however, the clustering result in a single subspace can be computed in polynomial time w.r.t. the input graph.

12.3.2 Finding clusters in different subspaces

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

By using a depth first search, the subspace S is analyzed before the subspace $S' = S \cup \{d\}$ for $d > \max\{S\}$. Based on the anti-monotonicity we know that each cluster in S' has to be a subset of a cluster in S. Thus, in subspace S' we do not have to start with the enriched subgraph G_S^V but it is sufficient to start with the vertex sets of the known clusters, i.e. if the clusters of subspace S are given by $Clus = \{C_1, \ldots, C_m\}$, we will determine the fixpoint $f(f(\ldots f(\{(O_i, S')\})))$ for each cluster $C_i = (O_i, S) \in Clus$. This is far more efficient since the vertex sets are smaller, i.e. Theorem 12.6 and 12.7 are now based on the set O_i instead of V. In Algorithm 12.1 the depth first traversal based on the clusters of the previous subspace is shown in line 21, 23 and 29.

The actual detection of clusters based on the vertices O is realized in line 13-19, which corresponds to the fixed point iteration described in the previous section. In our algorithm, we use the following observation to calculate the fixed point: If $\mathcal{C}$ is the largest fixed point of the function f_S, then obviously any individual cluster $C_i \in \mathcal{C}$ is also a fixed point, i.e. $f_S(\{C_i\}) = \{C_i\}$. Thus, if a graph G_S^O contains *a single* $(minPts{-}1)$-core O_1 with $O_1 = O$, we have reached a fixed point for

Algorithm 12.1: The DB-CSC algorithm

method: $main()$
1 $Result = \emptyset$ // current result set
2 $queue = \emptyset$ // priority queue with clusters and subtrees, descendingly sorted by quality
3 **for** $d \in Dim$ **do** $DFS_traversal(\{d\}, V, \emptyset)$
4 **while** $queue \neq \emptyset$ **do**
5 remove first (highest-quality) object Obj from $queue$
6 **if** Obj *is cluster* **then** // check redundancy
7 **for** $C \in Result$ **do** **if(** $Obj \prec_{red} C$**)** **goto** line 4 // discard redundant cluster
8 $Result = Result \cup \{Obj\}$ // cluster is non-redundant
9 **else** // Obj is subtree $ST = (S, O, Q_{max}, Parents, Parents_{red})$
10 **if** $Parents_{red} \cap Result \neq \emptyset$ **then** **goto** line 4 // discard whole subtree
11 **else** $DFS_traversal(S, O, Parents)$ // subtree is non-redundant, restart traversal
12 **return** $Result$

method: $DFS_traversal$(subspace S, candidate vertices O, parent clusters $Parents$)
13 $foundClusters = \emptyset$, $prelimClusters = \{O\}$
14 **while** $prelimClusters \neq \emptyset$ **do**
15 remove first candidate O_x from $prelimClusters$
16 generate enriched subgraph $G_S^{O_x}$
17 determine $(minPts-1)$-cores $\rightarrow Cores = \{O'_1, \ldots, O'_m\}$
18 **if** $|Cores| = 1 \wedge O'_1 = O_x$ **then** $foundClusters = foundClusters \cup \{(O'_1, S)\}$
19 **else** $prelimClusters = prelimClusters \cup Cores$
20 add $foundClusters$ to $queue$
21 **for** $C_i = (O_i, S) \in foundClusters$ **do**
22 $Parents_i = Parents \cup \{C_i\}$
23 **for** $d \in \{\max\{S\}+1, \ldots, D\}$ **do**
24 determine $Parents_{red} \subseteq Parents_i$ to which whole subtree is redundant
25 **if** $Parents_{red} \neq \emptyset$ **then**
26 calc. subtree information $ST = (S \cup \{d\}, O_i, Q_{max}, Parents_i, Parents_{red})$
27 add ST to $queue$ // (currently) do not traverse subtree!
28 **else**
29 $DFS_traversal(S \cup \{d\}, O_i, Parents_i)$ // check only subsets of O_i

this individual cluster. In this case, we can add the cluster to the set of detected clusters (line 18). In the other cases, we recursively have to repeat the procedure for each $(minPts-1)$-core $\{O_1, \ldots, O_m\}$ detected in G_S^Q (line 19). Though, instead of applying f_S to the whole set of cores simultaneously, we just have to apply it for each core individually (line 15), until all clusters are detected.

Using a depth first search enables us to store a set of parent clusters (beforehand detected in lower dimensional subspaces) that a new cluster is based on (cf. line 22). Furthermore, given a set of vertices O in the subspace S we know that by traversing the current subtree only clusters of the kind $C_{reach} = (O_{reach}, S_{reach})$

with $O_{reach} \subseteq O$ and $S \subseteq S_{reach} \subseteq S \cup \{\max\{S\}+1, \ldots, D\}$ can be detected. This information together with the redundancy model allows a further speed-up of the algorithm. The overall aim is to stop the traversal of a subtree if each of the reachable (potential) clusters is redundant to one parent cluster, i.e. if there exists $C \in Parents$ such that $C_{reach} \prec_{red} C$ for each C_{reach}. Traversing such a subtree is not worthwhile since the contained clusters are probably excluded from the result later on due to their redundancy.

Redundancy of a subtree occurs if the three properties introduced in Definition 12.6 hold. The second property (object overlap) is always fulfilled since each O_{reach} is a subset of any cluster from $Parents$ (cf. anti-monotonicity). The maximal possible quality of the clusters C_{reach} can be estimated by $Q_{max} = |O| \cdot |S \cup \{\max\{S\}+1, \ldots, D\}|$. By focusing on the clusters $C_p = (O_p, S_p) \in Parents$ with $Q_{max} < Q(C_p)$ we ensure the first redundancy property. The third property (dimension overlap) is ensured if $|S_p| \geq |S \cup \{\max\{S\}+1, \ldots, D\}| \cdot r_{dim}$ holds. In this case we get for each C_{reach}: $|S_p| \geq |S_{reach}| \cdot r_{dim} \Leftrightarrow |S_p \cap S_{reach}| \geq |S_{reach}| \cdot r_{dim} \Leftrightarrow \frac{|S_p \cap S_{reach}|}{|S_{reach}|} \geq r_{dim}$. Those parent clusters fulfilling all three properties are stored within $Parents_{red}$ (line 24).

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

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

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

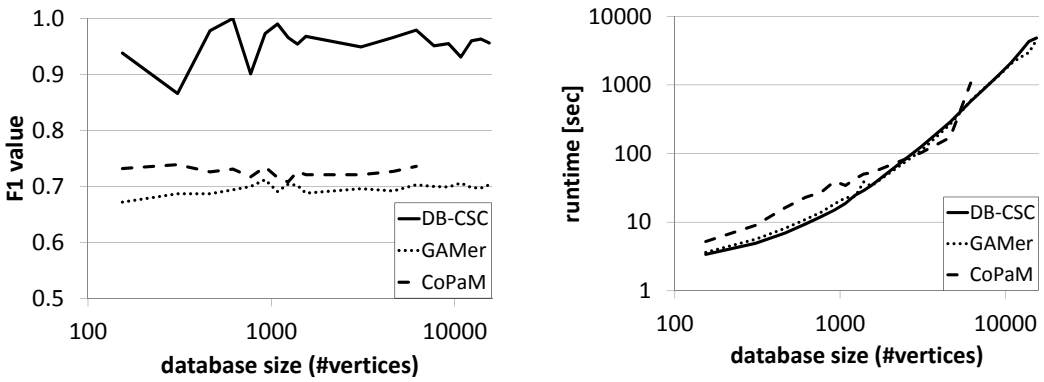


Figure 12.8: Quality (left) and runtime (right) w.r.t. varying database size

12.4 Experimental Analysis

Setup. We compare DB-CSC to GAMER and CoPaM [MCRE09]. In our experiments we use real world data sets as well as synthetic data. By default the synthetic datasets have 20 attribute dimensions and contain 80 combined clusters each with 15 nodes and 5 relevant dimensions. Additionally we add random nodes and edges to represent noise in the data. The clustering quality is measured by the F1 measure as also done in the previous chapters of this thesis. The efficiency is measured by the algorithms' runtime.

Varying characteristics of the data. In the first experiment (Figure 12.8) we vary the database size of our synthetic datasets by varying the number of generated combined clusters. The runtime of all algorithms increases with increasing database size (please note the logarithmic scale on both axes). For the datasets with more than 7000 vertices, CoPaM is not applicable any more due to heap overflows (4GB). While the runtimes of the different algorithms are very similar, in terms of clustering quality DB-CSC obtains significantly better results than the other approaches. The competing approaches tend to output only subsets of the hidden clusters due to their restrictive cluster models.

In the next experiment (Figure 12.9) we vary the dimensionality of the hidden clusters. The runtime of all algorithms increases for higher dimensional clusters. The clustering qualities of DB-CSC and CoPaM slightly decrease. This can be explained by the fact that for high dimensional clusters it is likely that additional clusters occur in subsets of the dimensions. However, DB-CSC still has the best clustering quality and runtime in this experiment.

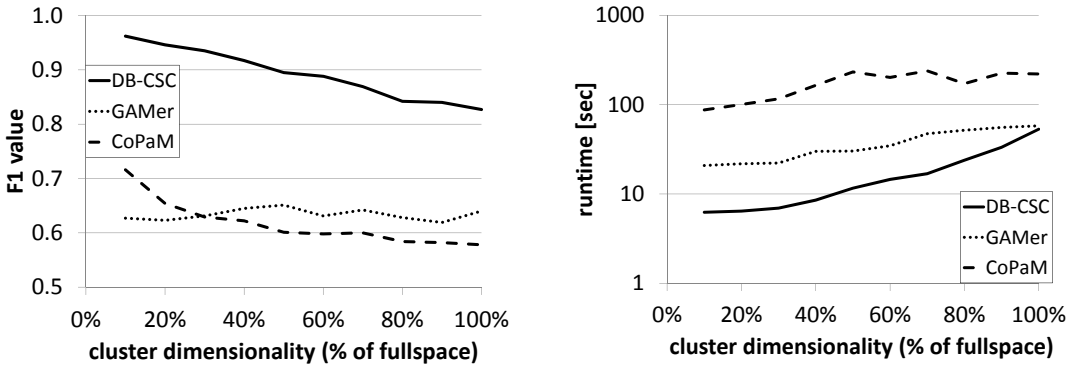


Figure 12.9: Quality (left) and runtime (right) w.r.t. varying dimensionality

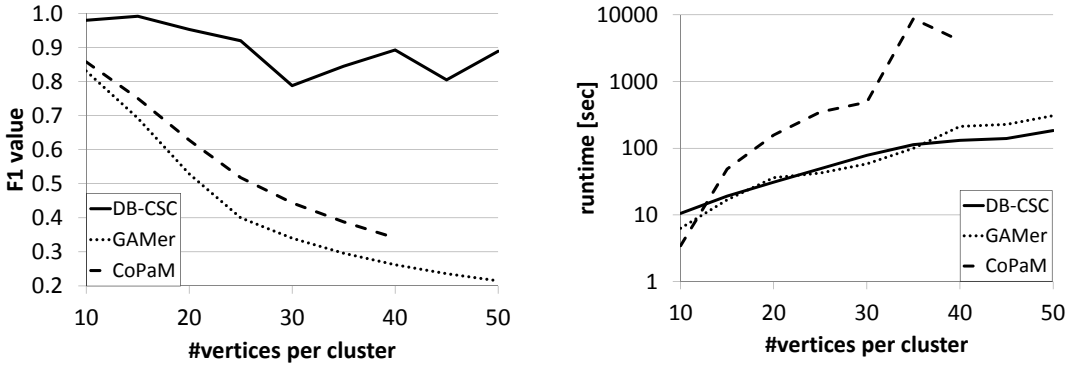
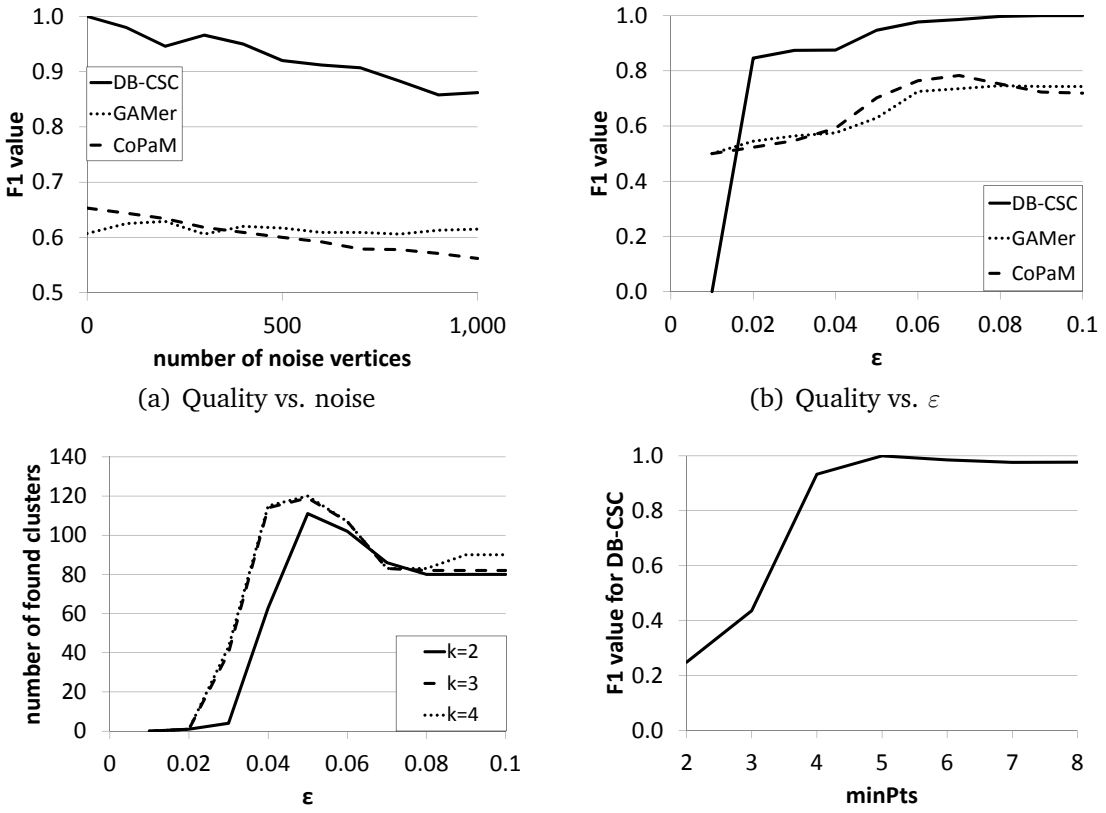


Figure 12.10: Quality (left) and runtime (right) w.r.t. varying cluster size

In Figure 12.10 the cluster size (i.e. the number of vertices per cluster) is varied. The runtimes of DB-CSC and GAMER are very similar to each other, whereas the runtime of CoPaM increases dramatically until it is not applicable any more. The clustering quality of DB-CSC remains relatively stable while the qualities of the other approaches decrease constantly. For increasing cluster sizes the expansion of the clusters in the graph as well as in the attribute space increases, thus the restrictive cluster models of GAMER and CoPaM can only detect subsets of them.

Robustness. In Figure 12.11(a) we analyze the robustness of the methods w.r.t. the number of "noise" vertices in the datasets. The clustering quality of all approaches decreases for noisy data, however the quality of DB-CSC is still reasonably high even for 1000 noise vertices (which is nearly 50% of the overall dataset).

In the next experiment (Figure 12.11(b)) we vary the clustering parameter ε, for a fixed value of $k = 2$. For GAMER and CoPaM we vary the allowed width of a cluster in the attribute space instead of ε. As shown in the figure, by choosing ε too



(a) Quality vs. noise

(b) Quality vs. ε

(c) Number of found clusters vs. ε & k

(d) Quality vs. $minPts$

Figure 12.11: Robustness of the methods w.r.t. noise and parameter values

small we cannot find all clusters and thus get smaller clustering qualities. However, for $\varepsilon > 0.05$ the clustering quality of DB-CSC remains stable. The competing methods have lower quality.

To further evaluate the detected clusterings, we depict in Figure 12.11(c) the number of found clusters for varying ε-values and different values for the parameter k. The synthetic dataset consists of 80 hidden clusters. For $k = 1$, DB-CSC does not detect any clusters, so only the results for $k = 2, 3, 4$ are included. For $k = 3$, DB-CSC finds slightly more clusters than for $k = 2$. However, by increasing the parameter further to $k = 4$ the number of clusters remains nearly stable. As seen before, if ε is chosen too small DB-CSC cannot find all the clusters and so outputs only small numbers of clusters. For $\varepsilon = 0.05$, more than the hidden 80 clusters are found, which can be explained by the fact that subsets of the hidden clusters can form additional clusters in higher dimensionalities. However, for even larger

ε-values the number of found clusters remains stable and all but only the hidden clusters are found.

In the last experiment (Figure 12.11(d)) we evaluate the robustness of DB-CSC w.r.t. the parameter $minPts$. For too small values for $minPts$, many vertex sets are falsely detected as clusters, thus we obtain small clustering qualities. However, for sufficiently high $minPts$ values the quality remains relatively stable, similar to the previous experiment.

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

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

12.5 Conclusion

In this chapter we introduced an integrated clustering model which simultaneously considers graph data and attribute data in subspaces. Our model is the first approach that exploits the advantages of density-based clustering in both domains. Based on the novel notion of local densities, our clusters correspond to dense regions in the graph as well as in the attribute space. To avoid redundancy in the result, our model selects only the most interesting clusters for the final clustering.

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

We developed the algorithm DB-CSC, which uses a fixed point iteration method to efficiently determine the combined clustering solution. The correctness, convergence, and complexity of this iteration method are proven analytically. In the experimental analysis we demonstrate the high clustering quality and the efficiency of our DB-CSC method.

With this chapter we conclude the third part of this thesis. We introduced integrated clustering techniques that simultaneously analyze network data in combination with vector data. Our methods avoid the problems of full-space clustering by analyzing subspace projections of the attribute data. We tackle the challenge of redundancy by proposing clustering models that confine the clustering result to only the interesting and non-redundant clusters. Overall, our methods enable an effective subspace clustering for one widely processed type of heterogeneous data.

Part IV

Summary

Chapter 13

Conclusion and Future Work

In this thesis, we proposed novel models for an effective subspace clustering of complex data. We analyzed three different types of data: vector data, imperfect data, and network data in combination with vector data. For each of these different data sources, we showed the importance of subspace clustering and the need to develop enhanced mining models. In this section, we briefly review the main research results and we present an outlook of possible future research directions.

13.1 Conclusion

In the first part of this thesis, we introduced novel subspace clustering models handling vector data. As a major contribution, all models tackled the challenge of redundancy in subspace clustering results. In our RESCU approach, we proposed a global optimization method to select the most-interesting and non-redundant clusters. Generalizing the weighted Set Cover problem, our model is able to detect overlapping subspace clusters without including redundant information. We proved the NP-hardness of our model and we developed an efficient algorithm determining an approximate clustering solution. Our model has been published at the 2009 IEEE International Conference on Data Mining [MAG+09b]. As a second contribution, our OSCLU model additionally tackles the challenge of multiple views induced by different subspace projections. In our OSCLU model we proposed a global optimization of the overall clustering exploiting the clusters' similarities regarding their sets of objects as well as their subspace projections. We proved the complexity of our optimization technique and we developed an efficient algorithm. The OSCLU model has been presented at the 2009 ACM Confer-

ence on Information and Knowledge Management [GMFS09]. An extension of this model to detect alternative clustering solutions based on previously known groupings was proposed in [GFMS10]. A different principle to detect multiple clustering solutions is exploited by our MVGen approach. We introduced a novel generative model to represent data showing multiple groupings in subspace projections. In contrast to traditional mixture models, our novel approach is not restricted to a single grouping and it allows to distinguish between relevant and irrelevant dimensions of clusters by performing Bayesian model selection. Our method has been published at the 2012 ACM SIGKDD Conference on Knowledge Discovery and Data Mining [GFS12]. As our next contribution, we proposed the paradigm of subspace correlation clustering. Our SSCC model detects local correlations in subspace projections of the data. By excluding collinear and induced correlations, our approach avoids redundant clusters and enables the detection of multiple overlapping correlations. The SSCC model has been presented at the 2012 ACM SIGKDD Conference on Knowledge Discovery and Data Mining [GFVS12]. Furthermore, as a general contribution, we provided an overview of different models for the detection of multiple clustering solutions in our tutorial 'Discovering Multiple Clustering Solutions: Grouping Objects in Different Views of the Data'. Our tutorial was presented at the 2010 IEEE International Conference on Data Mining [MGFS10], the 2011 SIAM International Conference on Data Mining [MGFS11], the 2012 IEEE International Conference on Data Engineering [MGFS12a], and the 2012 Pacific-Asia Conference on Knowledge Discovery and Data Mining [MGFS12b].

In the second part of this thesis, we proposed subspace clustering techniques handling imperfect data. As our first contribution, we developed an integrated clustering method that copes with the problem of missing values. Our model for fault tolerant subspace clustering allows multiple instantiations of missing values based on the considered subspace projection and it adapts to the clusters' characteristics. We introduced novel pruning techniques to realize an efficient execution of our algorithm. The experiments demonstrated the robustness of our method against a high amount of missing values. Our model has been published at the 2011 IEEE International Conference on Data Mining [GMRS11]. As a second contribution, we introduced a method to perform subspace clustering on uncertain data. We presented different variants of subspace cluster definitions and introduced techniques to realize overlapping subspace clusters in uncertain data. We developed an efficient algorithm, and we showed in experiments that our integrated clustering technique outperforms competing methods. This work has been

presented at the 2010 SIAM International Conference on Data Mining [GKS10].

In the third part of this thesis, we introduced subspace clustering techniques handling network data in combination with vector data. As our first contribution, the GAMER method joins the paradigms of subspace clustering and dense subgraph mining. Our GAMER method avoids redundant information but still allows multiple overlapping clusters in network data. We proposed several pruning techniques to realize an efficient integrated mining of such combined data sources. In experiments, we showed the quality and efficiency of our GAMER method. Our model has been published at the 2010 IEEE International Conference on Data Mining [GFBS10]. As a second contribution, we introduced the EDCAR model exploiting a global optimization of the overall clustering to select the most-interesting and non-redundant clusters. This novel principle increases the quality of the overall clustering solution. We proved the complexity of our method, and we developed an efficient algorithm to determine an approximate solution based on the GRASP principle. As our last contribution of this part, we proposed the first density-based clustering method simultaneously analyzing network data and vector data. Our DB-CSC model exploits the novel principle of local density computation to avoid misleading clusters and to achieve the advantages of density-based clustering for this specific type of heterogeneous data. We showed that a fixed point iteration can be utilized to determine the clustering result, and we proved to correctness, convergence, and complexity of this approach. The DB-CSC method has been presented at the 2011 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases [GBS11].

Overall, the proposed techniques enable an effective clustering in subspace projections of complex data. In several experiments, we demonstrated that existing techniques are outperformed by our novel methods. Furthermore, based on these methods and the insights gained in this thesis, additional research results were achieved. The application of subspace clustering for the purpose of indexing was shown in [GKLS11b], and the novel principle of tracing subspace clusters in temporal data was introduced in [GKLS11a, GKLS12]. As a general contribution to the research community, we developed the OpenSubspace toolkit [MAG+09a, MAG+11] providing a variety of subspace clustering techniques and allowing the comparability of experimental results. Focusing on the analysis of multiple clustering solutions, we furthermore developed extensions [GFKS10, GKFS10] of the OpenSubspace toolkit that allow an interactive exploration of the detected clustering results. These demonstrations were presented at the 2010 International

Conference on Very Large Data Bases and the 2010 IEEE International Conference on Data Mining. To also support the application of subspace clustering in practical applications, we proposed in [GKS11] an extension of the PMML standard providing a baseline to describe, exchange, and process subspace clustering results.

13.2 Future Work

Based on the models and results presented in this thesis, further challenging research questions arise. Incorporating prior knowledge into the clustering task is a promising direction. Often, the user is aware of certain knowledge that can be used to guide the clustering process. For example, constraints which objects must belong to the same cluster might be given or the user might already know some groupings and is only interested in alternative solutions. Exploiting this information in novel subspace clusterings techniques can enhance the quality of the detected clustering solutions.

Handling imperfect data in subspace clustering methods is still an ongoing research topic. Generative models, as, for example, used in our MVGen approach, have the potential to directly incorporate imperfect information, and suitable extensions of this model can be investigated. Furthermore, while in thesis we introduced methods to analyze incomplete and uncertain information, subspace clustering models for other kinds of imperfect data are not available so far but are worth to be developed in the future.

Also for the area of subspace clustering in network data, several further interesting research objectives arise. In this work, we focused on graphs with unlabeled relations; however, in some applications different types of edges or even edges annotated with feature vectors might occur. Designing novel models to handle this kind of information is a challenge that should be investigated in the future. Additionally, the problem of imperfect information has to be tackled for subspace clustering in network data. Since, e.g., missing edges occur frequently, combining the insights gained in the second and third part of this thesis is a promising research direction.

Finally, the paradigm of subspace clustering can be extended to further complex data types: relational data, time series data, and dynamic network data represent only a small selection of data types for which the huge potential of subspace clustering is worth to be investigated.

Appendices

Bibliography

[ABD+08] Elke Achtert, Christian Böhm, Jörn David, Peer Kröger, and Arthur Zimek. Global correlation clustering based on the hough transform. *Statistical Analysis and Data Mining (SADM)*, 1(3):111–127, 2008.

[ABK+06] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Deriving quantitative models for correlation clusters. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 4–13, 2006.

[ABK+07a] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. On exploring complex relationships of correlation clusters. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, page 7, 2007.

[ABK+07b] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Robust, complete, and efficient correlation clustering. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 413–418, 2007.

[ABKZ06] Elke Achtert, Christian Böhm, Peer Kröger, and Arthur Zimek. Mining hierarchies of correlation clusters. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 119–128, 2006.

[ACHM02] Marc Aerts, Gerda Claeskens, Niel Hens, and Geert Molenberghs. Local multiple imputation. *Biometrika*, 89(2):375, 2002.

[AGGR98] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages

94–105, 1998.

[AGGR05] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery Journal (DMKD)*, 11:5–33, 2005.

[AKMS07] Ira Assent, Ralph Krieger, Emmanuel Müller, and Thomas Seidl. DUSC: Dimensionality unbiased subspace clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 409–414, 2007.

[AKMS08a] Ira Assent, Ralph Krieger, Emmanuel Müller, and Thomas Seidl. EDSC: Efficient density-based subspace clustering. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1093–1102, 2008.

[AKMS08b] Ira Assent, Ralph Krieger, Emmanuel Müller, and Thomas Seidl. Inscy: Indexing subspace clusters with in-process-removal of redundancy. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 719–724, 2008.

[AKZ08] Elke Achtert, Hans-Peter Kriegel, and Arthur Zimek. Elki: A software system for evaluation of subspace clustering algorithms. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 580–585. 2008.

[AR10] Mohammad S. Aziz and Chandan K. Reddy. A robust seedless algorithm for correlation clustering. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 28–37, 2010.

[ARS02] James Abello, Mauricio G. C. Resende, and Sandra Sudarsky. Massive quasi-clique detection. In *Proceedings of the Latin American Theoretical Informatics Symposium (LATIN)*, pages 598–612, 2002.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 487–499, 1994.

[AW10] Charu C. Aggarwal and Haixun Wang. *Managing and Mining Graph Data*. Springer, New York, 2010.

[AWY+99] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 61–72, 1999.

[AY00] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 70–81, 2000.

[AY09] Charu C. Aggarwal and Philip S. Yu. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 21(5):609–623, 2009.

[BB06] Eric Bae and James Bailey. Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 53–62, 2006.

[BBC04] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[Bes86] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986.

[BGRS99] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the International Conference on Database Theory (ICDT)*, pages 217–235, 1999.

[Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer, New York, 2006.

[BKG+05] Arindam Banerjee, Chase Krumpelman, Joydeep Ghosh, Sugato Basu, and Raymond J. Mooney. Model-based overlapping clustering. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery*

and Data Mining (KDD), pages 532–537, 2005.

[BKKK04] Christian Böhm, Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. Density connected clustering with local subspace preferences. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 27–34, 2004.

[BKKZ04] Christian Böhm, Karin Kailing, Peer Kröger, and Arthur Zimek. Computing clusters of correlation connected objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 455–466, 2004.

[BM98] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Annual Conference on Learning Theory (COLT)*, pages 92–100, 1998.

[BP96] Patrick Bosc and Henri Prade. An introduction to the fuzzy set and possibility theory-based treatment of flexible queries and uncertain or imprecise databases. In *Uncertainty Management in Information Systems*, pages 285–324. 1996.

[BRB06] Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In *Proceedings of the International Conference on Conceptual Structures (ICCS)*, pages 144–157, 2006.

[Bre02] Richard P. Brent. *Algorithms for minimization without derivatives*. Dover Publications Inc., 2002.

[BS04] Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 19–26, 2004.

[BT78] Richard J. Beckman and Garth L. Tietjen. Maximum likelihood estimation for the beta distribution. *Journal of Statistical Computation and Simulation*, 7(3-4):253–258, 1978.

[BT85] Piero P. Bonissone and Richard M. Tong. Editorial: Reasoning with uncertainty in expert systems. *International Journal of Man-Machine Studies*, 22(3):241–250, 1985.

[BV11] Angela Bonifati and Yannis Velegrakis. Schema matching and mapping: from usage to evaluation. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 527–529, 2011.

[BZ07] Björn Bringmann and Albrecht Zimmermann. The chosen few: On identifying valuable patterns. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 63–72, 2007.

[CB10] Nicolas Cebron and Michael R. Berthold. Active learning in parallel universes. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1621–1624, 2010.

[CCKN06] Michael Chau, Reynold Cheng, Ben Kao, and Jackey Ng. Uncertain data mining: An example in clustering location data. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 199–204, 2006.

[CCWA02] Jacob Cohen, Patricia Cohen, Stephen G. West, and Leona S. Aiken. *Applied multiple regression/correlation analysis for the behavioral sciences*. Lawrence Erlbaum Assoc Inc., 3rd edition, 2002.

[CFD07] Ying Cui, Xiaoli Z. Fern, and Jennifer G. Dy. Non-redundant multi-view clustering via orthogonalization. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 133–142, 2007.

[CFZ99] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 84–93, 1999.

[Chv79] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[CKP03] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 551–562, 2003.

[CM08] Graham Cormode and Andrew McGregor. Approximation algorithms

for clustering uncertain data. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 191–200, 2008.

[DB10a] Xuan Hong Dang and James Bailey. Generation of alternative clusterings using the cami approach. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 118–129, 2010.

[DB10b] Xuan Hong Dang and James Bailey. A hierarchical information theoretic technique for the discovery of non linear alternative clusterings. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 573–582, 2010.

[DGM06] Sergey N. Dorogovtsev, Alexander V. Goltsev, and Jose F. F. Mendes. K-core organization of complex networks. *Physical Review Letters*, 96(4):40601, 2006.

[DHZ+01] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 107–114, 2001.

[DQ08] Ian Davidson and Zijie Qi. Finding alternative clusterings using constraints. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 773–778, 2008.

[DS07] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

[DWP+07] Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu. Community detection in large-scale social networks. In *Proceedings of the WebKDD/SNA-KDD workshop in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 16–25, 2007.

[EGG+06] Martin Ester, Rong Ge, Byron J. Gao, Zengjian Hu, and Boaz Ben-Moshe. Joint cluster analysis of attribute data and relationship data: the connected k-center problem. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 246–257, 2006.

[EIV07] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S.

Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(1):1–16, 2007.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.

[FA10] Andrew Frank and Arthur Asuncion. UCI machine learning repository, http://archive.ics.uci.edu/ml, 2010.

[FB08] Qiang Fu and Arindam Banerjee. Multiplicative mixture models for overlapping clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 791–796, 2008.

[FW06] Rudolf J. Freund and William Wilson. *Regression analysis: statistical modeling of a response variable*. Academic Press, 2006.

[GBS11] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. DB-CSC: A density-based approach for subspace clustering in graphs with feature vectors. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 565–580, 2011.

[GFBS10] Stephan Günnemann, Ines Färber, Brigitte Boden, and Thomas Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 845–850, 2010.

[GFKS10] Stephan Günnemann, Ines Färber, Hardy Kremer, and Thomas Seidl. CoDA: Interactive cluster based concept discovery. *PVLDB*, 3(2):1633–1636, 2010.

[GFM+11] Stephan Günnemann, Ines Färber, Emmanuel Müller, Ira Assent, and Thomas Seidl. External evaluation measures for subspace clustering. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1363–1372, 2011.

[GFMS10] Stephan Günnemann, Ines Färber, Emmanuel Müller, and Thomas Seidl. ASCLU: Alternative subspace clustering. In *Proceedings of the*

International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust) in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2010.

[GFS12] Stephan Günnemann, Ines Färber, and Thomas Seidl. Multi-view clustering using mixture models in subspace projections. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2012.

[GFVS12] Stephan Günnemann, Ines Färber, Kittipat Virochsiri, and Thomas Seidl. Subspace correlation clustering: Finding locally correlated dimensions in subspace projections of the data. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2012.

[GH06] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3), 2006.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP- Completeness*. W. H. Freeman, San Francisco, 1979.

[GKFS10] Stephan Günnemann, Hardy Kremer, Ines Färber, and Thomas Seidl. MCExplorer: Interactive exploration of multiple (subspace) clustering solutions. In *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1387–1390, 2010.

[GKLS11a] Stephan Günnemann, Hardy Kremer, Charlotte Laufkötter, and Thomas Seidl. Tracing evolving clusters by subspace and value similarity. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 444–456, 2011.

[GKLS11b] Stephan Günnemann, Hardy Kremer, Dominik Lenhard, and Thomas Seidl. Subspace clustering for indexing high dimensional data: A main memory index based on local reductions and individual multi-representations. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 237–248, 2011.

[GKLS12] Stephan Günnemann, Hardy Kremer, Charlotte Laufkötter, and Thomas Seidl. Tracing evolving subspace clusters in temporal cli-

mate data. In *Data Mining and Knowledge Discovery Journal (DMKD)*, volume 24, pages 387–410, 2012.

[GKS10] Stephan Günnemann, Hardy Kremer, and Thomas Seidl. Subspace clustering for uncertain data. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 385–396, 2010.

[GKS11] Stephan Günnemann, Hardy Kremer, and Thomas Seidl. An extension of the PMML standard to subspace clustering models. In *Proceedings of the Workshop on Predictive Model Markup Language (PMML) in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 48–53, 2011.

[GMFS09] Stephan Günnemann, Emmanuel Müller, Ines Färber, and Thomas Seidl. Detection of orthogonal concepts in subspaces of high dimensional data. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1317–1326, 2009.

[GMRS11] Stephan Günnemann, Emmanuel Müller, Sebastian Raubach, and Thomas Seidl. Flexible fault tolerant subspace clustering for data with missing values. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 231–240, 2011.

[HCS+07] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, Jérémy Besson, and Mohammed J. Zaki. Origami: Mining representative orthogonal graph patterns. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 153–162, 2007.

[HH07] Rave Harpaz and Robert M. Haralick. Mining subspace correlations. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 335–342, 2007.

[HK98] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 58–65, 1998.

[HK01] Jiawei Han and Micheline Kamber. *Data mining: Concepts and techniques*. Morgan Kaufmann, San Francisco, 2001.

[HZZL02] Daniel Hanisch, Alexander Zien, Ralf Zimmer, and Thomas Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18:145–154, 2002.

[ITM06] Ryutaro Ichise, Hideaki Takeda, and Taichi Muraki. Research community mining with topic identification. In *Proceedings of the International Conference on Information Visualisation (IV)*, pages 276–281, 2006.

[JL07] Svante Janson and Malwina J. Luczak. A simple solution to the k-core problem. *Random structures & algorithms*, 30(1-2):50–62, 2007.

[Jol02] Ian T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[KKK04] Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 246–257, 2004.

[KKKW03] Karin Kailing, Hans-Peter Kriegel, Peer Kröger, and Stefanie Wanka. Ranking interesting subspaces for clustering high dimensional data. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 241–252, 2003.

[KKPS04] Karin Kailing, Hans-Peter Kriegel, Alexey Pryakhin, and Matthias Schubert. Clustering multi-represented objects with noise. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 394–403, 2004.

[KKRW05] Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Sebastian Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 250–257, 2005.

[KKZ09] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1–58, 2009.

[KMS03] Jeremy Kubica, Andrew W. Moore, and Jeff G. Schneider. Tractable group detection on large link data sets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 573–576, 2003.

[KP05a] Hans-Peter Kriegel and Martin Pfeifle. Density-based clustering of uncertain data. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 672–677, 2005.

[KP05b] Hans-Peter Kriegel and Martin Pfeifle. Hierarchical density-based clustering of uncertain data. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 689–692, 2005.

[KWX+06] Eamonn J. Keogh, Li Wei, Xiaopeng Xi, Sang-Hee Lee, and Michail Vlachos. LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 882–893, 2006.

[LHSY07] Jiuyong Li, Xiaodi Huang, Clinton Selke, and Jianming Yong. A fast algorithm for finding correlation clusters in noise data. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 639–647, 2007.

[LR87] Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data*. Wiley, New York, 1987.

[LW08] Guimei Liu and Limsoon Wong. Effective pruning techniques for mining quasi-cliques. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 33–49, 2008.

[LWZY06] Bo Long, Xiaoyun Wu, Zhongfei (Mark) Zhang, and Philip S. Yu. Unsupervised learning on k-partite graphs. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 317–326, 2006.

[LZY07] Bo Long, Zhongfei (Mark) Zhang, and Philip S. Yu. A probabilistic framework for relational clustering. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 470–479, 2007.

[MAG+09a] Emmanuel Müller, Ira Assent, Stephan Günnemann, Timm Jansen, and Thomas Seidl. OpenSubspace: An open source framework for evaluation and exploration of subspace clustering algorithms in weka. In *Proceedings of the Open Source in Data Mining Workshop (OSDM 2009) in conjunction with Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 2–13, 2009.

[MAG+09b] Emmanuel Müller, Ira Assent, Stephan Günnemann, Ralph Krieger, and Thomas Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 377–386, 2009.

[MAG+11] Emmanuel Müller, Ira Assent, Stephan Günnemann, Patrick Gerwert, Matthias Hannen, Timm Jansen, and Thomas Seidl. A framework for evaluation and exploration of clustering algorithms in subspaces of high dimensional databases. In *Proceedings of the GI Conference on Database Systems for Business, Technology, and the Web (BTW)*, pages 347–366, 2011.

[MAGS11] Emmanuel Müller, Ira Assent, Stephan Günnemann, and Thomas Seidl. Scalable density-based subspace clustering. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1077–1086, 2011.

[MAK+09] Emmanuel Müller, Ira Assent, Ralph Krieger, Stephan Günnemann, and Thomas Seidl. DensEst: Density estimation for data mining in high dimensional spaces. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 173–184, 2009.

[MCRE09] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining cohesive patterns from graphs with feature vectors. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 593–604, 2009.

[MGAS09] Emmanuel Müller, Stephan Günnemann, Ira Assent, and Thomas Seidl. Evaluating clustering in subspace projections of high dimensional data. *PVLDB*, 2(1):1270–1281, 2009.

[MGFS10] Emmanuel Müller, Stephan Günnemann, Ines Färber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at the IEEE International Conference on Data Mining (ICDM)*, page 1220, 2010.

[MGFS11] Emmanuel Müller, Stephan Günnemann, Ines Färber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at the SIAM International Conference on Data Mining (SDM)*, 2011.

[MGFS12a] Emmanuel Müller, Stephan Günnemann, Ines Färber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at the IEEE International Conference on Data Engineering (ICDE)*, 2012.

[MGFS12b] Emmanuel Müller, Stephan Günnemann, Ines Färber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2012.

[MS08] Gabriela Moise and Jörg Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 533–541, 2008.

[MSE06] Gabriela Moise, Jörg Sander, and Martin Ester. P3C: A robust projected clustering algorithm. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 414–425, 2006.

[NAJ04] Jennifer Neville, Micah Adler, and David Jensen. Spectral Clustering with Links and Attributes. Technical report, Department of Computer Science, University of Massachusetts, Amherst, 2004.

[NE10] Xuan Vinh Nguyen and Julien Epps. minCEntropy: A novel information theoretic approach for the generation of alternative clusterings. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 521–530, 2010.

[NGC01] Harsha Nagesh, Sanjay Goil, and Alok Choudhary. Adaptive grids for

clustering massive data sets. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2001.

[NKC+06] Wang Kay Ngai, Ben Kao, Chun Kit Chui, Reynold Cheng, Michael Chau, and Kevin Y. Yip. Efficient clustering of uncertain data. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 436–445, 2006.

[Par96] Simon Parsons. Current approaches to handling imperfect information in data and knowledge bases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 8(3):353–372, 1996.

[Pas97] Vangelis Th. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys*, 29(2):171–209, 1997.

[PB05] Ruggero G. Pensa and Jean-François Boulicaut. Towards fault-tolerant formal concept analysis. In *Proceedings of the Congress of the Italian Association for Artificial Intelligence (AI*IA)*, pages 212–223, 2005.

[PG09] Ardian Kristanto Poernomo and Vivekanand Gopalkrishnan. Towards efficient mining of proportional fault-tolerant frequent itemsets. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 697–706, 2009.

[PHL04] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6(1):90–105, 2004.

[PJAM02] Cecilia Magdalena Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 418–427, 2002.

[PJZ05] Jian Pei, Daxin Jiang, and Aidong Zhang. On mining cross-graph quasi-cliques. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 228–238, 2005.

[PM06] Anne Patrikainen and Marina Meila. Comparing subspace cluster-

ings. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(7):902–916, 2006.

[PR02] Leonidas S. Pitsoulis and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. In *Handbook of Applied Optimization*, pages 168–182, New York, 2002. Oxford University Press.

[QD09] Zijie Qi and Ian Davidson. A principled and flexible framework for finding alternative clusterings. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 717–726, 2009.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[Res09] Mauricio G. C. Resende. Greedy randomized adaptive search procedures. In *Encyclopedia of Optimization*, pages 1460–1469. 2nd edition, 2009.

[RR10] Mauricio G. C. Resende and Celso C. Ribeiro. Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications. In *Handbook of Metaheuristics (International Series in Operations Research & Management Science)*, pages 283–320. 2nd edition, 2010.

[Rub87] Donald B. Rubin. *Multiple imputation for nonresponse in surveys*, volume 99. Wiley, New York, 1987.

[Rym92] Ron Rymon. Search through systematic set enumeration. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 539–550, 1992.

[RZ07] Jianhua Ruan and Weixiong Zhang. An efficient spectral algorithm for network community discovery and its applications to biological and social networks. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 643–648, 2007.

[S+05] R. Shyamsundar et al. A DNA microarray survey of gene expression in normal human tissues. *Genome Biology*, 6, 2005.

[S+06] C. Stark et al. BioGRID: a general repository for interaction datasets.

Nucleic acids research, 34, 2006.

[Sch78] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[Sch97] Joseph L. Schafer. *Analysis of incomplete multivariate data*, volume 72. Chapman & Hall/CRC, 1997.

[SEKX98] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery Journal (DMKD)*, 2(2):169–194, 1998.

[SG02a] Joseph L. Schafer and John W. Graham. Missing data: Our view of the state of the art. *Psychological methods*, 7(2):147, 2002.

[SG02b] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[Sil86] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

[SJR08] Manas Somaiya, Christopher M. Jermaine, and Sanjay Ranka. Learning correlations using the mixture-of-subsets model. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(4), 2008.

[SJR10] Manas Somaiya, Christopher M. Jermaine, and Sanjay Ranka. Mixture models for learning low-dimensional roles in high-dimensional data. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 909–918, 2010.

[SJZ10] Arlei Silva, Wagner Meira Jr., and Mohammed J. Zaki. Structural correlation pattern mining for large graphs. In *Proceedings of the Workshop on Mining and Learning with Graphs (MLG) in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 119–126, 2010.

[ST96] Abraham Silberschatz and Alexander Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 8(6):970–974, 1996.

[STM07] Motoki Shiga, Ichigaku Takigawa, and Hiroshi Mamitsuka. A spectral clustering approach to optimally combining numericalvectors with a modular network. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 647–656, 2007.

[SZ04] Karlton Sequeira and Mohammed J. Zaki. Schism: A new approach for interesting subspace mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 186–193, 2004.

[Tzv90] Arie Tzvieli. Possibility theory: An approach to computerized processing of uncertainty. *Journal of the American Society for Information Science and Technology (JASIST)*, 41(2):153–154, 1990.

[US07] I. Ulitsky and R. Shamir. Identification of functional modules using network topology and high-throughput data. *BMC Systems Biology*, 1(1), 2007.

[WB07] Bernd Wiswedel and Michael R. Berthold. Fuzzy clustering in parallel universes. *International Journal of Approximate Reasoning*, 45(3):439–454, 2007.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition, 2005.

[WHB10] Bernd Wiswedel, Frank Höppner, and Michael R. Berthold. Learning in parallel universes. *Data Mining and Knowledge Discovery Journal (DMKD)*, 21(1):130–152, 2010.

[WXC09] Junjie Wu, Hui Xiong, and Jian Chen. Adapting the right measures for k-means clustering. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 877–886, 2009.

[WZZ06] Jianyong Wang, Zhiping Zeng, and Lizhu Zhou. Clan: An algorithm for mining closed cliques from large dense graph databases. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 73–82, 2006.

[XL08] Huajie Xu and Guohui Li. Density-based probabilistic clustering of uncertain data. *Proceedings of the International Conference on Com-*

puter Science and Software Engineering (CSSE), 4:474–477, 2008.

[XYFS07] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. SCAN: a structural clustering algorithm for networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 824–833, 2007.

[YCN05] Kevin Y. Yip, David W. Cheung, and Michael K. Ng. On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 329–340, 2005.

[YFB01] Cheng Yang, Usama Fayyad, and Paul S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 194–203, 2001.

[YM03] Man Lung Yiu and Nikos Mamoulis. Frequent-pattern based iterative projected clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 689–692, 2003.

[YM05] Man Lung Yiu and Nikos Mamoulis. Iterative projected clustering by subspace mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(2):176–189, 2005.

[YWWY02] Jiong Yang, Wei Wang, Haixun Wang, and Philip S. Yu. delta-clusters: Capturing subspace correlation in a large data set. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 517–528, 2002.

[ZCY09] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.

[ZCY10] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Clustering large attributed graphs: An efficient incremental approach. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 689–698, 2010.

[ZFW+03] Barry R. Zeeberg, Weimin Feng, Geoffrey Wang, May D. Wang, Anthony T. Fojo, Margot Sunshine, Sudarshan Narasimhan, David W.

Kane, William C. Reinhold, Samir Lababidi, Kimberly J. Bussey, Joseph Riss, J. C. Barrett, and John N. Weinstein. GoMiner: a resource for biological interpretation of genomic and proteomic data. *Genome Biology*, 4(4), 2003.

[ZPW08a] Xiang Zhang, Feng Pan, and Wei Wang. CARE: Finding local linear correlations in high dimensional data. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 130–139, 2008.

[ZPW08b] Xiang Zhang, Feng Pan, and Wei Wang. Redus: finding reducible subspaces in high dimensional data. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 961–970, 2008.

[ZWZK06] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 797–802, 2006.

[ZWZK07] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. Out-of-core coherent closed quasi-clique mining from large dense graph databases. *ACM Transactions on Database Systems (TODS)*, 32(2):13, 2007.

[ZYH+07] Feida Zhu, Xifeng Yan, Jiawei Han, Philip S. Yu, and Hong Cheng. Mining colossal frequent patterns by core pattern fusion. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 706–715, 2007.

[ZYL09] Shijie Zhang, Jiong Yang, and Shirong Li. RING: An Integrated Method for Frequent Representative Subgraph Mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 1082–1087, 2009.

List of Publications

[AGKS09] Ira Assent, Stephan Günnemann, Hardy Kremer, and Thomas Seidl. High-dimensional indexing for multimedia features. In *Proceedings of the GI Conference on Database Systems for Business, Technology, and the Web (BTW)*, pages 187–206, 2009.

[AKGS10] Ira Assent, Hardy Kremer, Stephan Günnemann, and Thomas Seidl. Pattern detector: fast detection of suspicious stream patterns for immediate reaction. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 709–712, 2010.

[AMG+10] Ira Assent, Emmanuel Müller, Stephan Günnemann, Ralph Krieger, and Thomas Seidl. Less is more: Non-redundant subspace clustering. In *Proceedings of the International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust) in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[BGHS12] Brigitte Boden, Stephan Günnemann, Holger Hoffmann, and Thomas Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2012.

[FGK+10] Ines Färber, Stephan Günnemann, Hans-Peter Kriegel, Peer Kröger, Emmanuel Müller, Erich Schubert, Thomas Seidl, and Arthur Zimek. On using class-labels in evaluation of clusterings. In *Proceedings of the International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust) in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[GBS11] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. DB-CSC: A density-based approach for subspace clustering in graphs with fea-

ture vectors. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 565–580, 2011.

[GBS12] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. Substructure clustering: A novel mining paradigm for arbitrary data types. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, 2012.

[GFBS10] Stephan Günnemann, Ines Färber, Brigitte Boden, and Thomas Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 845–850, 2010.

[GFKS10] Stephan Günnemann, Ines Färber, Hardy Kremer, and Thomas Seidl. CoDA: Interactive cluster based concept discovery. *PVLDB*, 3(2):1633–1636, 2010.

[GFM+11] Stephan Günnemann, Ines Färber, Emmanuel Müller, Ira Assent, and Thomas Seidl. External evaluation measures for subspace clustering. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1363–1372, 2011.

[GFMS10] Stephan Günnemann, Ines Färber, Emmanuel Müller, and Thomas Seidl. ASCLU: Alternative subspace clustering. In *Proceedings of the International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust) in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[GFS12] Stephan Günnemann, Ines Färber, and Thomas Seidl. Multi-view clustering using mixture models in subspace projections. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2012.

[GFVS12] Stephan Günnemann, Ines Färber, Kittipat Virochsiri, and Thomas Seidl. Subspace correlation clustering: Finding locally correlated dimensions in subspace projections of the data. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2012.

[GKFS10] Stephan Günnemann, Hardy Kremer, Ines Färber, and Thomas Seidl. MCExplorer: Interactive exploration of multiple (subspace) clustering solutions. In *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1387–1390, 2010.

[GKLS11a] Stephan Günnemann, Hardy Kremer, Charlotte Laufkötter, and Thomas Seidl. Tracing evolving clusters by subspace and value similarity. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 444–456, 2011.

[GKLS11b] Stephan Günnemann, Hardy Kremer, Dominik Lenhard, and Thomas Seidl. Subspace clustering for indexing high dimensional data: A main memory index based on local reductions and individual multi-representations. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 237–248, 2011.

[GKLS12] Stephan Günnemann, Hardy Kremer, Charlotte Laufkötter, and Thomas Seidl. Tracing evolving subspace clusters in temporal climate data. In *Data Mining and Knowledge Discovery Journal (DMKD)*, volume 24, pages 387–410, 2012.

[GKS10] Stephan Günnemann, Hardy Kremer, and Thomas Seidl. Subspace clustering for uncertain data. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 385–396, 2010.

[GKS11] Stephan Günnemann, Hardy Kremer, and Thomas Seidl. An extension of the PMML standard to subspace clustering models. In *Proceedings of the Workshop on Predictive Model Markup Language (PMML) in conjunction with the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 48–53, 2011.

[GMFS09] Stephan Günnemann, Emmanuel Müller, Ines Färber, and Thomas Seidl. Detection of orthogonal concepts in subspaces of high dimensional data. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1317–1326, 2009.

[GMRS11] Stephan Günnemann, Emmanuel Müller, Sebastian Raubach, and Thomas Seidl. Flexible fault tolerant subspace clustering for data with missing values. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 231–240, 2011.

[GS10] Stephan Günnemann and Thomas Seidl. Subgraph mining on directed and weighted graphs. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 133–146, 2010.

[KGFS10] Philipp Kranen, Stephan Günnemann, Sergej Fries, and Thomas Seidl. MC-Tree: Improving bayesian anytime classification. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 252–269, 2010.

[KGHS12] Hardy Kremer, Stephan Günnemann, Arne Held, and Thomas Seidl. Mining of temporal coherent subspace clusters in multivariate time series databases. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2012.

[KGI+11] Hardy Kremer, Stephan Günnemann, Anca Maria Ivanescu, Ira Assent, and Thomas Seidl. Efficient processing of multiple DTW queries in time series databases. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 150–167, 2011.

[KGS10] Hardy Kremer, Stephan Günnemann, and Thomas Seidl. Detecting climate change in multivariate time series data by novel clustering and cluster tracing techniques. In *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 96–97, 2010.

[MAG+09a] Emmanuel Müller, Ira Assent, Stephan Günnemann, Timm Jansen, and Thomas Seidl. OpenSubspace: An open source framework for evaluation and exploration of subspace clustering algorithms in weka. In *Proceedings of the Open Source in Data Mining Workshop (OSDM 2009) in conjunction with Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 2–13, 2009.

[MAG+09b] Emmanuel Müller, Ira Assent, Stephan Günnemann, Ralph Krieger, and Thomas Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 377–386, 2009.

[MAG+11] Emmanuel Müller, Ira Assent, Stephan Günnemann, Patrick Gerwert, Matthias Hannen, Timm Jansen, and Thomas Seidl. A framework for evaluation and exploration of clustering algorithms in subspaces of high dimensional databases. In *Proceedings of the GI Conference on Database Systems for Business, Technology, and the Web (BTW)*, pages 347–366, 2011.

[MAGS11] Emmanuel Müller, Ira Assent, Stephan Günnemann, and Thomas Seidl. Scalable density-based subspace clustering. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1077–1086, 2011.

[MAK+09] Emmanuel Müller, Ira Assent, Ralph Krieger, Stephan Günnemann, and Thomas Seidl. DensEst: Density estimation for data mining in high dimensional spaces. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 173–184, 2009.

[MGAS09] Emmanuel Müller, Stephan Günnemann, Ira Assent, and Thomas Seidl. Evaluating clustering in subspace projections of high dimensional data. *PVLDB*, 2(1):1270–1281, 2009.

[MGFS10] Emmanuel Müller, Stephan Günnemann, Ines Färber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at the IEEE International Conference on Data Mining (ICDM)*, page 1220, 2010.

[MGFS11] Emmanuel Müller, Stephan Günnemann, Ines Färber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at the SIAM International Conference on Data Mining (SDM)*, 2011.

[MGFS12a] Emmanuel Müller, Stephan Günnemann, Ines Färber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at the IEEE International Conference on Data Engineering (ICDE)*, 2012.

[MGFS12b] Emmanuel Müller, Stephan Günnemann, Ines Färber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2012.

Curriculum Vitae

Dipl.-Inform. Stephan Günnemann

Born on June 21, 1983
in 49716 Meppen, Germany
German Citizen *E-Mail*: guennemann@cs.rwth-aachen.de

Education

- ◇ since June 2008, **Doctoral studies** in Computer Science at the RWTH Aachen University
- ◇ October 2003 - May 2008, **Diplom** in Computer Science with Business Administration as a minor subject at the RWTH Aachen University
- ◇ June 2002, **Abitur** from Gymnasium Marianum Meppen

Professional Experience

- ◇ May 2011 - June 2011, School of Computing Science at Simon Fraser University, Vancouver, Canada: **Visiting researcher**
- ◇ since July 2008, Data Management and Data Exploration Group at RWTH Aachen University: **Research assistant**
- ◇ April 2007 - September 2007, Data Management and Data Exploration Group at RWTH Aachen University: **Student assistant**
- ◇ April 2006 - September 2006, Deutsche Post Endowed Chair of Optimization of Distribution Networks at RWTH Aachen University: **Student assistant**
- ◇ July 2002 - April 2003, Bürgerhilfe e.V. Kreisstelle Meppen: Mandatory **Civilian Service**
- ◇ July 2000 - August 2000 & April 2001 & June 2001 - August 2001, Competence Center Informatik GmbH in Meppen: **Working student**

Academic Honors

- ◇ ECML PKDD Best Paper Award for the paper “DB-CSC: A density-based approach for subspace clustering in graphs with feature vectors“, 2011
- ◇ Friedrich-Wilhelm-Preis for diploma thesis on “Approximations for efficient subspace clustering in high-dimensional databases”, 2009
- ◇ Springorum-Denkmünze for Diplom in Computer Science with overall mark ’excellent’, 2009
- ◇ Schöneborn-Preis for the best Vordiplom (Bachelor degree equivalent) of the year in Computer Science at the RWTH Aachen University, 2006