Fast and Effective Methods for Pattern Mining in Complex Data

Sebastian Göbl

Munich 2022

Fast and Effective Methods for Pattern Mining in Complex Data

Sebastian Göbl

Dissertation
im Fach Informatik
an der Fakultät für Mathematik,
Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Sebastian Göbl
aus München

München, den 26.10.2022

Erstgutachter: Prof. Dr. Christian Böhm
Zweitgutachter: Prof. Dr. Karsten Borgwardt
Tag der mündlichen Prüfung: 15.04.2024

Abstract

Among today's challenges in big data science, called the "Big V's", *variety* induces an enormous complexity of data. A vast number of portable or stationary devices collect sensor or user information of various types. In medicine, diagnostic tools observe many biological parameters and produce high-dimensional data for each patient. The patterns we are looking for are obscured by correlated attributes and noise. User information collected from the web (e.g., Facebook data) consists of very heterogeneous data domains: for example, age is provided as a natural number and height as a real number – both continuous ranges easily analyzed by distance functions. Heterogeneity is increased by adding dichotomous or general categorical domains (e.g., sex or citizenship) or relational information (e.g., Facebook likes). Very large, complex networks, such as email or authorship connections, are approached using the abstraction of graph representation. Graph mining methods search for patterns and rules that allow us to understand how these networks were created. In realistic settings, such as communication or power networks, graphs exhibit heterogeneous patterns beyond simple clusters or communities. The goal of this thesis is to contribute to tackling the challenges that arise from the complexity of data.

Challenges: We take on *five key challenges* (C 1 to 5) derived from mining complex data and pointed out by the data mining community: *graph and network mining* **(C 1)** requires new efficient and effective algorithms for pattern mining due to the complexity of graphs over classic dimensional data, considering edge types, edge weights, or edge numbers following power laws. *Explainable data mining* **(C 2)** is required for transparent output and transparent algorithmic decisions since increasing model complexity makes it harder for the user to understand and learn the structure and patterns in the data. *High-dimensional data mining* **(C 3)** asks for efficient algorithms to tackle the large data size, increasing noise, and the large number of dimensions. *Mining heterogeneous data types* **(C 4)**

requires integrated solutions for pattern mining of data with not only numerical but heterogeneous data domains. *Parameter-free data mining* **(C 5)** automatically finds suitable parameters for algorithms since complex data and models make it tedious for the user to define good input parameters.

Contributions: We present four fast and effective algorithms to address these challenges. Our first two proposals, MeGS and Spectral Lens, focus on graph mining, while our third and fourth methods, FOSSCLU and INTEGRATE, focus on clustering. Our graph mining algorithm *MeGS* (Partitioning Meaningful Subgraph Structures using Minimum Description Length) looks for homogeneous patterns (subgraph structures) in graphs. It allows a complete understanding of these structures and, thus, the whole graph (C 1). It matches meaningful primitives (clique, hub, tree, bipartite and sparse subgraphs) to all partitions of a graph. Every node of the graph is part of precisely one structure and has an interpretable context (C 2). MeGS is a fast and parameter-free split-and-merge algorithm that automatically finds the optimal structures that achieve the best compression by minimum description length (MDL) (C 5). With weighted and unweighted, directed and undirected networks, the graph challenge (C 1) can take on more complex shapes: With *Spectral Lens*, we introduce an efficient and effective method to address even very large (C 3) graphs, despite having additional edge weights or directed edges (C 4). Our algorithm identifies connectivity patterns given by a dictionary of rules to gain insights (C 2) into a graph's communities, e.g., their power-law distribution, and discovers anomalies. Spectral Lens can automatically find the optimal number of communities (C 5) that describe the network best. Our algorithm *FOSSCLU* (Finding the Optimal Subspace for Clustering) takes up the challenge of data dimensionality (C 3) in clustering. For human understanding, it is important to simplify and categorize things. Therefore, it is highly valuable to find clusters in one common low-dimensional subspace where we can study not only the intra-cluster but also the inter-cluster relationships (C 2). We propose FOSSCLU to find this subspace in a joint, alternating process of clustering and dimensionality reduction. FOSSCLU is rendered parameter-free with the aid of MDL (C 5). We approach the difficulties in clustering heterogeneous data (C 4) by proposing *INTEGRATE*: Our parameter-free algorithm integrates the information supplied by heterogeneous numerical and categorical dimensions and exploits both for clustering. INTEGRATE is efficient and scalable and automatically decides on parameter values (C 5) using MDL.

Our four algorithms address key challenges arising from complex data. Our methods are evaluated by extensive experiments on real-world data and provide efficient and effective solutions to contribute to the key tasks of data mining.

Zusammenfassung

Eine der aktuellen Herausforderungen des Big-Data-Science, die auch als "Big V's" bezeichnet werden, ist die Vielfalt der Daten (engl.: *variety*), die zu einer enormen Komplexität führt. Sehr viele tragbare oder stationäre Geräte erfassen unterschiedlichste Sensor- oder Benutzerinformationen. In der Medizin überwachen Diagnosegeräte viele biologische Parameter und erzeugen für jeden einzelnen Patienten hochdimensionale Datensätze. Korrelierte Attribute und Rauschen erschweren das Auffinden relevanter Muster. Im Internet gesammelte Nutzerinformationen, beispielsweise Daten bei Facebook, umfassen sehr heterogene Datendomänen: so wird etwa das Alter als natürliche Zahl und die Körpergröße als reelle Zahl angegeben — beides sind kontinuierliche Wertebereiche, die sich leicht mit Distanzfunktionen analysieren lassen. Die Heterogenität nimmt zu, wenn dichotome oder allgemein kategorische Daten (wie Geschlecht oder Staatsangehörigkeit) oder auch relationale Informationen wie *Likes* bei Facebook hinzukommen. Sehr große und komplexe Netzwerke – wie E-Mail-Verbindungen oder Publikationsnetzwerke – lassen sich gut mit Hilfe von Graphabstraktion untersuchen. Die Methoden des Graph-Minings suchen nach Mustern und Regeln um zu verstehen, wie Netzwerke entstanden sind. In der Realität zeigen Graphen (z.B. Kommunikationsnetzwerke oder Stromversorgungsnetze) heterogene Strukturen, die komplexer sind als einfache *Cluster* oder *Communities*. Die vorliegende Arbeit soll einen Beitrag zur Bewältigung der Herausforderungen leisten, die sich aus der Komplexität der Daten ergeben.

Herausforderungen: Wir befassen uns mit *fünf zentralen Herausforderungen* (C 1 to 5) des Minings komplexer Daten, die in der Data-Mining-Community besonders hervorgehoben werden: Das *Graph- und Netzwerk-Mining* **(C 1)** erfordert effiziente und effektive Algorithmen zur Mustersuche, da Graphen deutlich komplexer als klassische, dimensionsbasierte Daten sind. Das liegt an Kantentypen, Kantengewichten und auch daran, dass die Anzahl der Kanten eines Graphen Potenzgesetzen folgen kann. Das *erklärbare Data-Mining* **(C 2)** ist nötig für transparente Ergebnisse und transparente Entscheidungsprozesse der Algorithmen, da der Anwender wegen der komplexen Modelle Strukturen und Muster in den Daten nur schwer nachvollziehen kann. Das *hochdimensionale Data-Mining* **(C 3)** erfordert effiziente Algorithmen für die großen Datenmengen mit starkem Rauschen und hoher Dimensionalität. Das *Mining heterogener Datentypen* **(C 4)** verlangt nach integrierten Lösungen zur Mustererkennung von Daten aus nicht nur numerischen, sondern auch heterogenen Domänen. Das *parameterfreie Data-Mining* **(C 5)** findet selbstständig geeignete Eingabeparameter für Algorithmen, da es die Komplexität der Daten und Modelle dem Anwender oft erschwert, selbst gute Parameter festzulegen.

Beiträge: Wir stellen vier schnelle und effektive Algorithmen vor, die einen Beitrag zur Bewältigung dieser Herausforderungen leisten. Unsere ersten beiden Methoden, MeGS und Spectral Lens, konzentrieren sich auf das Graph-Mining, die anderen beiden, FOSSCLU und INTEGRATE, auf das Clustering. Unser Graph-Mining-Algorithmus *MeGS* (Partitionierung aussagekräftiger Subgraphstrukturen unter Verwendung der Minimum-Description-Length, engl.: *Partitioning Meaningful Subgraph Structures using Minimum Description Length*) sucht nach homogenen Mustern (Subgraphstrukturen) in Graphen und ermöglicht ein umfassendes Verständnis dieser Strukturen und damit des ganzen Graphen (C 1). Allen Partitionen eines Graphen werden aussagekräftige Primitive zugeordnet (Clique, Hub, Baum, bipartite und dünn besetzte Subgraphen). Jeder Knoten eines Graphen gehört zu genau einer Struktur und erhält dadurch einen interpretierbaren Kontext (C 2). MeGS ist ein schneller, parameterfreier Split-and-Merge-Algorithmus; mit Hilfe des Prinzips der Minimum-Description-Length (MDL) findet er automatisch diejenige Partitionierung, die zur besten Komprimierung des Graphen führt (C 5). Durch Graphen mit und ohne Kantengewichten und durch gerichtete und ungerichtete Kanten erhöht sich die Komplexität im Graph-Mining weiter (C 1): Mit *Spectral Lens* schaffen wir eine effiziente und effektive Möglichkeit, um auch sehr große Graphen (C 3) trotz zusätzlicher Kantengewichte und gerichteter Kanten (C 4) zu verarbeiten. Unser Algorithmus identifiziert Konnektivitätsmuster über ein Verzeichnis von Regeln; dadurch erhält er Einblick (C 2) in die Communities der Graphen, beispielsweise in ihre Potenzgesetzverteilung, und er bringt Anomalien zum Vorschein. Spectral Lens kann automatisch errechnen, durch wie viele Communities (C 5) ein Netzwerk am besten erfasst wird. Unser Algorithmus *FOSSCLU* (Finden des optimalen Unterraums für das Clustering, engl.: *Finding the Optimal Subspace for Clustering*) befasst sich mit der Herausforderung der Datendimensionalität (C 3) im Clustering. Für die menschliche Auffassung ist es wichtig, Dinge zu vereinfachen und zu kategorisieren. Daher ist es sehr hilfreich, Cluster in einem gemeinsamen, niedrig-dimensionalen Unterraum zu finden, wo wir Beziehungen innerhalb eines Clusters und auch zwischen den Clustern untersuchen können (C 2). Um diesen Unterraum in einem gemeinsamen, alternierenden Prozess aus Clustering und Dimensionalitätsreduktion zu finden, stellen wir FOSSCLU vor. Durch die Verwendung von MDL muss der Anwender keine Eingabeparameter festlegen (C 5). Den Schwierigkeiten im Clustering heterogener Daten (C 4) begegnen wir mit unserem Algorithmus *INTEGRATE*: unser parameterfreier Algorithmus integriert Daten mit numerischen und kategorischen Dimensionen, um beide gleichermaßen in das Clustering einzubeziehen. INTEGRATE ist effizient und skalierbar. Eingabeparameter werden mit Hilfe von MDL automatisch bestimmt (C 5).

Unsere vier Algorithmen nehmen zentrale Herausforderungen in Angriff, die durch die Komplexität der Daten verursacht werden. Unsere Methoden werden durch umfangreiche Experimente an Echtweltdaten evaluiert und leisten mit effizienten und effektiven Lösungen einen Beitrag zu den Kernaufgaben des Data-Minings.

To my family

Acknowledgments

This thesis could not have been written without the support of others to whom I would like to express my deepest and sincere gratitude.

First and foremost, I would like to express my warmest thanks to my supervisor, Professor Dr. Christian Böhm. He financed my research assistant position, my trips to conferences, and my research stay at Carnegie Mellon University. His enthusiasm for elegant algorithms and our fruitful discussions greatly inspired my work. His guidance and support were invaluable.

Likewise, I would like to express my gratitude to Professor Dr. Claudia Plant. She supported me with guidance, discussions, and reviews throughout my work. Her energy to conquer new scientific questions was a great motivation.

My sincere thanks also go to Professor Christos Faloutsos, Ph.D., who welcomed me to his group for a research stay and supported me with many valuable and fruitful discussions.

I am very thankful to Professor Dr. Karsten Borgwardt, who readily agreed to act as the second reviewer for this thesis.

For their collaboration and inspiration, I am grateful to my colleagues (in alphabetical order): Alexandra Derntl, Dr. Jing Feng, Dr. Xiao He, Dr. Nina Hubig, Professor Srijan Kumar, Ph.D., Dr. Son T. Mai, Dr. Sam Maurus, Dr. Dominik Mautz, Annika Tonch, Dr. Bianca Wackersreuther, Dr. Wei Ye, and Dr. Linfei Zhou.

I thank the following students who supported my work (in alphabetical order): Anna Beer, David Englmeier, and Martin Winter.

I am also very grateful to Susanne Grienberger and Franz Krojer for their administrative and technical help during my time as a research assistant.

Last but not least, I would like to express my deepest gratitude to my family for their love and support.

List of Publications

The following papers were published in peer-reviewed conferences. An asterisk (*) indicates that the authors contributed equally.

1. Christian Böhm*, Sebastian Goebl*, Annahita Oswald*, Claudia Plant*, Michael Plavinski*, and Bianca Wackersreuther*. Integrative Parameter-Free Clustering of Data with Mixed Type Attributes. In: *Advances in Knowledge Discovery and Data Mining.* PAKDD 2010, pp. 38-47.

2. Nora Broy*, Sebastian Goebl*, Matheus Hauder*, Thomas Kothmayr*, Michael Kugler*, Florian Reinhart*, Martin Salfer*, Kevin Schlieper*, Elisabeth André. "A Cooperative In-Car Game for Heterogeneous Players." In: *AutomotiveUI '11: Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications.* 2011, pp. 167–176.

3. Sebastian Goebl, Claudia Plant, Marc Lobbes, and Anke Meyer-Baese. Quantitative Analysis of Breast DCE-MR Images Based on ICA and an Empirical Model. In: *Proc. SPIE 8401, Independent Component Analyses, Compressive Sampling, Wavelets, Neural Net, Biosystems, and Nanoengineering X.* 2012, 84010Z.

4. Son T. Mai, Sebastian Goebl, Claudia Plant. "A Similarity Model and Segmentation Algorithm for White Matter Fiber Tracts." In: *ICDM '12: Proceedings of the 12th IEEE International Conference on Data Mining.* 2012, pp. 1014–1019.

5. Sebastian Goebl, Claudia Plant, Marc Lobbes, and Anke Meyer-Baese. CAD-System Based on Kinetic Analysis for Non-Mass-Enhancing Lesions

in DCE-MRI. In: *Proc. SPIE 8750, Independent Component Analyses, Compressive Sampling, Wavelets, Neural Net, Biosystems, and Nanoengineering XI.* 2013, 87500R.

6. Sebastian Goebl, Anke Meyer-Baese, Marc Lobbes, and Claudia Plant. Segmentation and Kinetic Analysis of Breast Lesions in DCE-MR Imaging Using ICA. In: *ITBAM '14: Information Technology in Bio- and Medical Informatics.* 2014, pp. 45–59.

7. Sebastian Goebl, Xiao He, Claudia Plant, and Christian Böhm. Finding the Optimal Subspace for Clustering. In: *ICDM 14: Proceedings of the IEEE 14th International Conference on Data Mining.* 2014, pp. 130-139.

8. David Englmeier, Nina Hubig, Sebastian Goebl, Christian Böhm. "Musical Similarity Analysis based on Chroma Features and Text Retrieval Methods." In: *BTW '15: Datenbanksysteme für Business, Technologie und Web Workshopband.* 2015, pp. 183–192.

9. Martin Winter, Sebastian Goebl, Nina Hubig, Christopher Pleines, Christian Böhm. "Development and Evaluation of a Facebook-based Product Advisor for Online Dating Sites." In: *BTW '15: Datenbanksysteme für Business, Technologie und Web Workshopband.* 2015, pp. 213-222.

10. Wei Ye, Sebastian Goebl, Claudia Plant, Christian Böhm. "FUSE: Full Spectral Clustering." In: *KDD 16: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2016, pp. 19851994.

11. Sebastian Goebl, Annika Tonch, Christian Böhm, and Claudia Plant. MeGS: Partitioning Meaningful Subgraph Structures using Minimum Description Length. In: *ICDM 16: Proceedings of the 16th IEEE International Conference on Data Mining.* 2016, pp. 889-894.

12. Sebastian Goebl, Srijan Kumar, and Christos Faloutsos. Spectral Lens: Explainable Diagnostics, Tools and Discoveries in Directed, Weighted Graphs. In: *ICDM 17: Proceedings of the 17th IEEE International Conference on Data Mining.* 2017, pp. 877-882.

The papers printed with **bold** numbers in the author's list of publications contribute to this thesis. The **papers 11** and **12** contribute to Part II (Fast and Effective Methods for Explainable Graph Structuring and Summarization), and the **papers 1** and **7** to Part III (Fast and Effective Methods for Parameter-free Clustering).

Contents

Part I.

Preliminaries

1. Introduction

In the past decade, analyzing data has become a part of mainstream society. Buzzwords like *data science*, *data analytics*, *big data*, *machine learning*, or *data-driven* have become familiar to many (Figure 1.1). Entire industries have sprung up around the analysis of data. Many fields of science and large parts of the economy analyze a wide variety of data in order to create surplus value and new knowledge. Due to the increasing number of use cases, the variety of data increases, too. The data come from a deep variety of sources, exhibit diverse structures, and come in high dimensions. Networks and relationships between data entities have gained particular attention. Since the establishment of the field of *data mining* over two decades ago, many of the classic data mining algorithms are either overchallenged or simply overwhelmed by the sheer volume of data. The increasing complexity of data has raised numerous new scientific questions in the field of data mining.

In this chapter, we outline the field of data mining with its core process of discovering knowledge in data. We shed light on different challenges caused by complexity in the data and hint at how we face those challenges with our contributions.

1.1. Data Mining and Knowledge Discovery in Databases

Facing large amounts of data but little information, the term *data mining* was coined over thirty years ago as a metaphor for digging up (useful) information from large amounts of data as gold is dug up in gold mining [HKP13]. Originally, data mining was considered a step in a larger process of processing data and

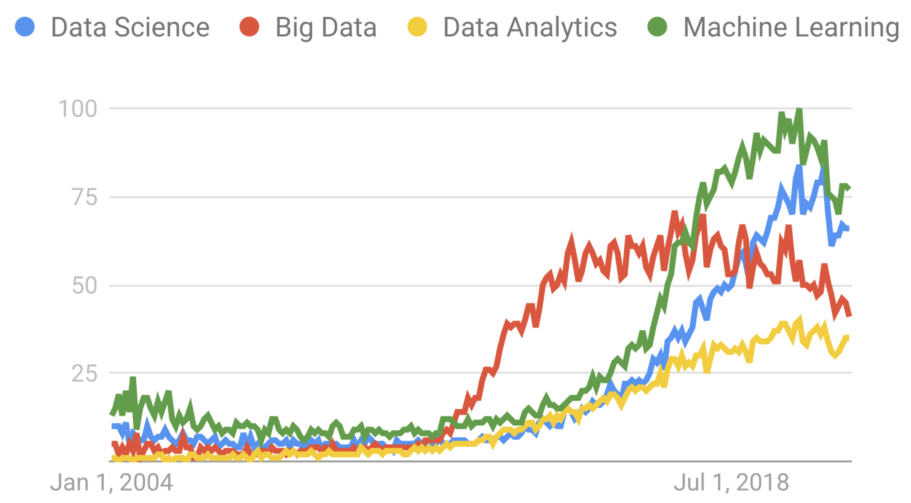


Figure 1.1.: Worldwide relative increase of buzzwords related to Data Science (2004–2020) in the Google web search. Percentage relative to the highest value (Figure courtesy of Google Trends [Goo21]).

extracting information from the data, named *knowledge discovery from databases (KDD)*. Fayyad, Piatetsky-Shapiro, and Smyth define [FPS96b]:

> *"**Knowledge discovery in databases** is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data."*

In five steps, the KDD process produces knowledge from data (Figure 1.2) [FPS96a]. We will go into the details of the KDD process below. Being one of these steps the term *data mining* is popularly used as a pars pro toto for the KDD process, and a modern definition is given by Han, Kamber, and Pei [HKP13]:

> *"**Data mining** is the process of discovering interesting patterns and knowledge from large amounts of data. The data sources can include databases, data warehouses, the Web, other information repositories, or data that are streamed into the system dynamically."*

This definition is also more advanced by including, e.g., streamed data and takes into account that data does not have to be stored in a structured format in a database.

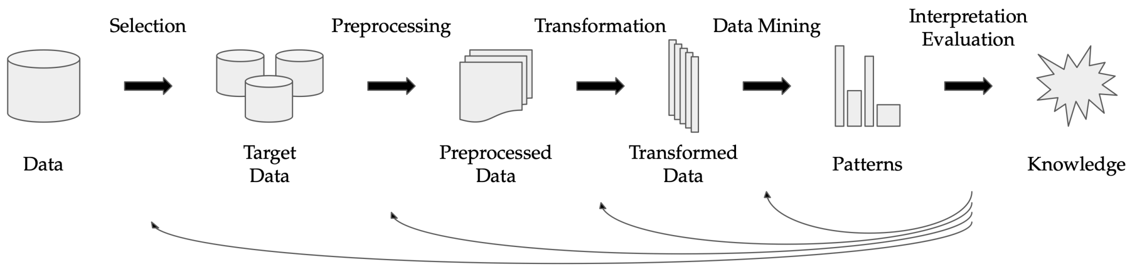


Figure 1.2.: The classic KDD process (sketched according to Figure 1 from [FPS96a]).

A downside of the term data mining is that it is a misleading metaphor. While gold mining implies digging through stone or mountains to discover gold, in contradistinction, data mining does not try to discover data but information or knowledge. Among the less ambiguous although less popular terms are *knowledge mining* or *pattern mining* — both a better analogy to the gold mining metaphor. This is the reason why the title of this thesis contains the term pattern mining instead. Here, we use *pattern* according to Fayyad, Piatetsky-Shapiro, and Smyth [FPS96a] as a pattern found in the data, a component of a model, e.g., a cluster in a clustering.

Distinction from Statistics, Machine Learning, and other Methods. Among the buzzwords of Figure 1.1 are *machine learning* and *big data*. How are they or the classic field of statistics related to data mining and KDD? — The KDD process is a multidisciplinary activity involving other fields, methods, and algorithms like statistics, machine learning, big data processing, information retrieval, pattern recognition, and many more [FPS96a; HKP13]. The data mining step of the KDD process applies a suitable method to extract patterns from the data.

The methods in the data mining step can be structured into *supervised* and *unsupervised* methods. Supervised methods or *supervised learning* learn from an attribute in the data to predict its outcome. This requires a training step and, thus, supervision. Instead, unsupervised methods like clustering do not need to be trained [Agg15].

1.1.1. Patterns in Data Mining

On a high level, we can separate the problems that data mining addresses into several categories. A common denominator in research is found in the following four divisions [FPS96a; Agg15; HKP13]: *association pattern mining*, *clustering*, *outlier detection*, and *classification*.

Association Pattern and Frequent Pattern Mining

A classic application is the shopping basket analysis, answering questions like *what is bought together?* or *who buys what?* More generally, *frequent patterns* or *frequent itemsets* are patterns observed frequently in data. An association rule defines that one or more attributes imply or are associated with another attribute. E.g., young adults that buy a notebook might buy software in the same order with a 50% chance (or *confidence*). Frequent pattern mining also relates to finding patterns or frequent substructures in graphs.

Clustering

Clustering or cluster analysis is the general approach to partition objects into groups or clusters (= sets of records) based on their similarity. The similarity is calculated by a similarity function (also called objective function or clustering function). Clustering can also be defined as an optimization problem, and the objective function maximizes the similarity between the records of the same cluster while minimizing the similarity of the records of different clusters. Often, only global optimizations are reached due to computational intensity. An example of a clustering problem is the partition of customers by age, income, gender, and location with the goal of determining the target audience for advertising.

Clustering is an unsupervised data mining technique that does not need labels to extract information from the data.

Outlier Detection

Outlier or anomaly detection searches the data for observations that deviate significantly from the other observations so that it is has to be explained by a different model. In cluster analysis, outliers show little similarity and cannot easily assigned to a cluster. Outlier analysis is also an unsupervised technique. A classic application is the detection of credit card fraud: outliers in the attributes *location*, *type of purchase*, *purchase amount and frequency* point to fraudulent activities and should be investigated.

Classification

For classification, a particular feature in the data, the class label, is used to train a model to predict this feature. In the training set, this feature is known. The trained model predicts this feature in data without a class label. Due to the requirement of a class label and a training set, this is a supervised technique. Classification is predicting a discrete attribute. For a continuous attribute, regression creates a model that predicts this feature.

1.1.2. Steps of the KDD Process

The KDD Process [FPS96a] consist of the following steps (Figure 1.2). The process is run iteratively, and the user can jump back to each step for refinement or a different choice of execution. E.g., different data can be selected, or another data mining algorithm can be chosen.

1. **Selection**: With domain expert knowledge and a goal in mind, a subset of attributes and records is chosen from available data sources. Unstructured and structured data is transferred in a joinable format as a prerequisite for further processing by ETL (extract, transform, and load) methods. The selection serves as target data for the next steps of the KDD process.

2. **Preprocessing**: The target data is cleaned by removing inconsistent data and evident noise. Strategies for handling missing attributes are chosen. E.g., a rolling average can fill missing values of a time series.

3. **Transformation**: The preprocessed data can be further reduced to select useful features and to reduce the dimensionality of the data. The user usually returns iteratively to this step to perform data mining on different

subsets of features. Transformation can also include projections of the data dimensions to a subspace.

4. **Data Mining**: A data mining method like one of the four major areas mentioned above is chosen and applied to the selected, preprocessed, and transformed data in order to extract patterns. This is the algorithmic core step and is used synonymously for the KDD process. Choosing and setting up an algorithm is not trivial and will often give results that do not allow the user to gain knowledge from the data.

5. **Interpretation and Evaluation**: The extracted patterns are interpreted, and the previous step will be repeated until information and knowledge is gained from the data. E.g., this can mean predicting new data or understanding the data by approximating a model that is able to generate the data. Visualizations of the extracted patterns are a valuable tool for human understanding. Finally, the newly gained knowledge is imported into another system for future use like presentation and decision making.

1.2. Challenges in Mining Complex Data

Several challenges have been named for research in data mining [HKP13; HG08; YW06]. Among these, the complexity of data causes difficulties that data mining algorithms have to manage. In the following, we derive five challenges that arise from handling complex data and are faced by the methods and algorithms proposed in this thesis.

- **Challenge 1: Graph and Network Mining** Graphs are a useful tool to model the relationships between entities and build network representations. In comparison to classic dimensional data, graphs are more complex: e.g., they can contain different edge types (directed or undirected links), the edge types can have weights, and edge numbers can follow power laws and grow rapidly. New algorithms need to be established to deal with graph data and their characteristics.

- **Challenge 2: Explainable Data Mining.** With increasing data complexity, the result of data mining algorithms tends to be less understandable. Often, a simple classification result is not enough as an output of the data mining process. Users need to gain insights into the underlying models that produce these results. *Black box algorithms* do not increase the knowledge about what causes algorithms to create clusters or classify data. Explainable data mining allows understanding algorithmic decisions and gives information about the decision-making process.
- **Challenge 3: High-Dimensional Data Mining.** With increasing dimensionality, the amount of noise spread over these dimensions usually increases and causes more complexity in the data. For techniques like clustering, the noise needs to be reduced while not losing information. This approach is an improvement over the classic KDD process that applies noise reduction before the data mining steps. Next to developing efficient algorithms to tackle the size of the data, the challenge is to interweave noise reduction and data mining algorithms.
- **Challenge 4: Mining Heterogeneous Data Types.** In real-world data, numerical and categorical data occurs in the same records. While numerical data allows using a distance function that is underlying to many clustering algorithms, records with mixed-type data require new algorithms. These methods should integrate data mining of both numerical and categorical data equally.
- **Challenge 5: Parameter-free Data Mining.** Especially with increasing complexity in the data, parameter settings are more challenging to decide. Internal measures that guide an algorithm to find a locally or globally optimal parameter setting take much work off the user's hands and might lead to results that would otherwise remain hidden.

1.3. Facing the Challenges with Our Contributions

In this thesis, we propose four data mining methods and algorithms to face the challenges that arise from complex data and that have been outlined in the previous section. The following overview explains how each of our contributions faces those challenges. With each contribution, we aim to provide both fast and effective solutions in order to be able to cope with high-dimensional data as well as results of superior quality. Our first two proposals of Part II cover the task of graph mining:

- **MeGS.** Our algorithm to find meaningful subgraph structures discovers patterns in *graphs*. It returns *meaningful* results that explain the graph so that the user gains insights not only about the similarity of nodes but also about all the underlying structures in a graph. MeGS uses the information-theoretic principle of minimum description length (MDL) that allows *automatic parametrization*.
- **Spectral Lens.** Our method Spectral Lens *explains large-scale graphs* by identifying nodes with similar connectivity patterns. It finds suspicious behavior in networks and is able to *automatically parametrize* by detecting the optimal number of groups.

In Part III, our proposed methods contribute to solving clustering problems:

- **FOSSCLU.** Our method FOSSCLU combines clustering and dimensionality reduction of *high-dimensional data* into one algorithm. Allowing to project the data to an arbitrary two-dimensional plane by a rigid orthonormal rotation of the data, it gives *visualizations to understand* the cluster structures. FOSSCLU uses MDL for *automatic parametrization*.
- **INTEGRATE.** Our algorithm INTEGRATE allows to cluster *heterogeneous* data with numerical and categorical attributes. Its objective function uses MDL for an integrated clustering function and for *automatic parametrization*.

1.4. Outline of this Thesis

This thesis is structured as follows: **Part I** contains this introduction followed by **Chapter 2** with a primer about the for this thesis most important and frequently used mathematical definitions together with an overview over our notation and a remark to the reproducibility of our results.

Part II presents our first two contributions to graph mining. In **Chapter 3** past work related to these methods is summarized. **Chapter 4** contains our proposed algorithm MeGS (Partitioning Meaningful Subgraph Structures using Minimum Description Length), and **Chapter 5** explains our second contribution to graph mining, our method Spectral Lens.

Part III contains our proposed clustering techniques, opening with a survey of past research related to our work in **Chapter 6**. Our algorithm FOSSCLU (Finding the Optimal Subspace for Clustering) to simultaneously performing clustering and dimensionality reduction on high-dimensional datasets is presented in **Chapter 7**. **Chapter 8** explains our algorithm INTEGRATE for integrated clustering of numerical and categorical attributes.

Part IV concludes this thesis with a summary and an outlook in **Chapter 9**.

2. Methodical and Mathematical Preliminaries

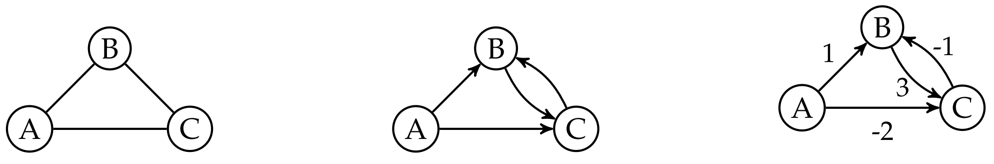
This chapter summarizes essential mathematical principles applied in this thesis for easier understanding. The focus here is on fundamental notations used across chapters. Section 2.1 summarizes graph-theoretical principles and notations important for following Part II of this thesis. In Section 2.2, we refresh the foundations of information-theoretic clustering with the minimum description length principle based on it and the external cluster-validity measures Dom and NMI. The minimum description length principle underlies methods from Part II and Part III of this thesis and the external cluster-validity measures (recapitulated in Section 2.2.2) for assessing our algorithmic results. Section 2.3 concludes the methodical preliminaries with notes on the notations used in this thesis.

2.1. Graphs

The following definitions give an overview of different types of graphs, characteristics of graphs, different subgraph structures, and graph representation in a matrix [Die17]. In this thesis, we use the term *network* synonymously for graphs, as a network can be represented by a graph.

Definition 2.1: (Undirected) Graph

An *(undirected) graph* is a pair $G = (V, E)$ of sets such that $E \subseteq \{\{v_i, v_j\} | v_i \in V \wedge v_j \in V\}$. The elements of V are the *nodes* (or *vertices*) and the elements of E the *edges* of graph G. We assume $V \cap E = \emptyset$.



(a) (Undirected) graph (b) Directed graph (c) Directed, weighted graph

Figure 2.1.: Directed, undirected and weighted graphs

An edge $\{v_i, v_i\} \in E$ with $v_i \in V$ is called a *self-loop*. Definition 2.1 allows only one edge between each pair of nodes. So-called *multiple edges*, i.e., more than one edge between a pair of nodes, are not allowed. Edges have no direction; therefore, we call the graph in Definition 2.1 an *undirected* graph. Figure 2.1a gives a graph $G = (V, E)$ with node set $V = \{A, B, C\}$ and edge set $E = \{\{A, B\}, \{A, C\}, \{B, C\}\}$. The number of nodes of a graph is the cardinality of V and the number of edges the cardinality of E. If unambiguous, we denote $n = |V|$ and $m = |E|$.

Definition 2.2: Directed Graph

A *directed graph* is a pair $G = (V, E)$ of sets such that $E \subseteq V \times V$. The elements of V are the nodes (or vertices) and the elements of E the edges of graph G. We assume $V \cap E = \emptyset$.

In a *directed* graph, an edge is represented by a tuple of nodes so that a maximum of two edges can exist between each pair of nodes. An edge (v_i, v_j) with $v_i, v_j \in V$ is *directed* from v_i to v_j. We define n and m as above. The undirected and directed graphs in Definitions 2.1 and 2.2 are called *unweighted*, as there is no additional attribute to describe the quality of an edge. In this sense, all edges have *equal weight*. Figure 2.1b depicts a directed graph $G = (V, E)$ with node set $V = \{A, B, C\}$ and edge set $E = \{(A, B), (A, C), (B, C), (C, B)\}$.

Definition 2.3: Weighted Graph

A *weighted graph* is a triplet $G = (V, E, w)$. V and E are defined as in Definition 2.1 (for an undirected, weighted graph) or as in Definition 2.2 (for a directed, weighted graph). The *weight function* $w : E \mapsto \mathbb{R} \setminus \{0\}$ defines a weight for every edge. We assume $V \cap E = \emptyset$.

An undirected or directed graph is extended to a *weighted* graph simply by adding the weight function w to the pair $G = (V, E)$. We allow negative weights for edges[1], but we disallow a zero edge weight; in this case, not drawing an edge is the preferred option. We define n and m as above. Note that unweighted graphs are simulated by weighted graphs with weight function $w : E \mapsto 1$. A directed, weighted graph $G = (V, E, w)$ with node set $V = \{A, B, C\}$, edge set $E = \{(A, B), (A, C), (B, C), (C, B)\}$ and weight function $w = \{((A, B), 1), ((A, C), -2), ((B, C), 3), ((C, B), -1)\}$ is given in Figure 2.1c.

Definition 2.4: Degree of a Node

The *degree* $d_G(v)$ of a node v of graph G is defined by $|\{\{v, v_j\} \in E\}|$ for undirected graphs and $|\{(v_j, v_j) \in E | v = v_i \vee v = v_j\}|$ for directed graphs.

Definition 2.5: Path, Cycle, Connected

A *path* is a graph $P = (V, E)$ with $V \neq \emptyset$ such that $V = \{v_1, v_2, \ldots, v_k\}$ and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}\}$ with $v_i, v_j \in V \wedge i \neq j \Rightarrow v_i \neq v_j$. The *length* of the path is $|E|$, and v_1, v_k are *linked* by the path.
A *cycle* is a graph $C = (V, E)$ such that $P = (V, E')$ is a path of length $|E'| \geq 2$ and $E = E' \cup \{v_k, v_1\}$.
An undirected graph $G = (V, E)$ (resp. $G = (V, E, w)$) is *acyclic* if there exists no cycle $C = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$.
A non-empty undirected graph G is *connected* if every pair of nodes in G is linked.

[1]Negative weights are a practical way to represent, e.g., negative ratings or opinions.

Definition 2.6: Subgraph

A *subgraph* G' of a graph $G = (V, E)$ (resp. $G = (V, E, w)$) is a graph $G' = (V', E')$ (resp. $G' = (V', E', w)$) such that $V' \subseteq V$ and $E' \subseteq E$.
An *induced subgraph* is a subgraph that contains all edges $(v_i, v_j) \in E$ (resp. $\{v_i, v_j\} \in E$) if $v_i \in V' \wedge v_j \in V'$.

The concept of *induced subgraphs* is essential for our studies on graph mining. If the nodes of a graph are divided into k disjoint non-empty subsets of V so that $V = \bigcup_{i=1}^{k} V_i$, we say that $V_1, \ldots, V_k$ form a *partition* of V. The nodes of each subset induce a subgraph of G so that all induced subgraphs together with the edges between each pair of induced subgraphs assemble the complete original graph. In our studies on graphs, we are interested in analyzing all nodes and edges in a graph. Hence, when using the term *subgraphs*, we refer to induced subgraphs from now on. Synonymously, we use the term *subgraph structures*.

To differentiate subgraph structures, we define several graph types which are known from graph theory. We give their definition in the following and an illustration in Figure 2.2.

Definition 2.7: Clique

A *clique* or a *complete graph* is a graph $G = (V, E)$ (resp. $G = (V, E, w)$) such that for all $v_i, v_j \in V$ $(i \neq j)$ there is an edge $(v_i, v_j) \in E$ (resp. $\{v_i, v_j\} \in E$).

Definition 2.8: Bipartite

A *bipartite* graph is a graph $G = (V, E)$ (resp. $G = (V, E, w)$) such that there is a partition into two sets of nodes V_1, V_2 so that for all edges $(v_i, v_j) \in E$ (resp. $\{v_i, v_j\} \in E$) with $i \neq j$ it holds that $v_i \in V_1 \Rightarrow v_j \in V_2$ and $v_i \in V_2 \Rightarrow v_j \in V_1$.

Definition 2.9: Hub

A *hub* is a graph $G = (V, E)$ (resp. $G = (V, E, w)$) with a distinguished node $v_i \in V$, the *hub node*, such that for all $v_j \in V$ it holds that $d_G(v_i) \gg d_G(v_j)$.

Definition 2.10: Dense, Sparse

A *dense* graph is a graph $G = (V, E)$ such that roughly $|E| \propto |V|^2$. A *sparse* graph is a graph $G = (V, E)$ such that roughly $|E| \propto |V|$. For small graphs, the definitions for spare and dense are only approximate.

For simplicity and because it is sufficient for our work, we restrict to undirected graphs in the following definition.

Definition 2.11: Forest, Tree, Root

A *forest* is an acyclic, undirected graph. A *tree* is a forest that is connected. A tree can have a distinguished node as the *root* of the tree.

A practical way to describe either type of graph in matrix notation, which allows applying rules of linear algebra, is given in the following.

Definition 2.12: Adjacency Matrix

The *adjacency matrix* $A = (a_{ij})_{n \times n}$ for graph G is defined

- if G is an unweighted graph (Definitions 2.1 and 2.2) as

$$a_{ij} := \begin{cases} 1 & if \{v_i, v_j\} \in E \\ 1 & if (v_i, v_j) \in E \\ 0 & otherwise \end{cases} \tag{2.1}$$

- if G is an weighted graph (Definition 2.3) as

$$a_{ij} := \begin{cases} w(\{v_i, v_j\}) & if \{v_i, v_j\} \in E \\ w((v_i, v_j)) & if (v_i, v_j) \in E \\ 0 & otherwise \end{cases} \tag{2.2}$$

For an undirected, bipartite graph, the adjacency matrix can also be defined as $A = (a_{ij})_{|V_1| \times |V_2|}$ with V_1 and V_2 being the disjoint sets of nodes of the graph such that there are only edges allowed from a node of one set to a node of the other set.

2.2. Information-Theoretic Clustering

For our graph partitioning method MeGS in Part II and for our clustering methods FOSSCLU and INTEGRATE in Part III, we use measures based on Claude E. Shannon's *Mathematical Theory of Communication* [Sha48], the fundamental paper establishing Information Theory. Clustering algorithms aim to arrange objects in clusters based on a similarity metric like a distance function. *More similar* objects are grouped in the same cluster, and *less similar* objects lie in different clusters. This is equivalent to increasing the order in each cluster. A cluster of similar objects contains less information than a cluster of very different objects.

Imagine a cluster of blue objects and a cluster of red objects. The first cluster gives and is described only by the information 'blue' and the other by 'red'. A cluster with red and blue objects is described by the information 'blue' and 'red'.

Shannon has introduced the principle of *entropy*, known from physics as a measure of information. In the example, the cluster with only red objects has a lower entropy than the cluster with red and blue objects.

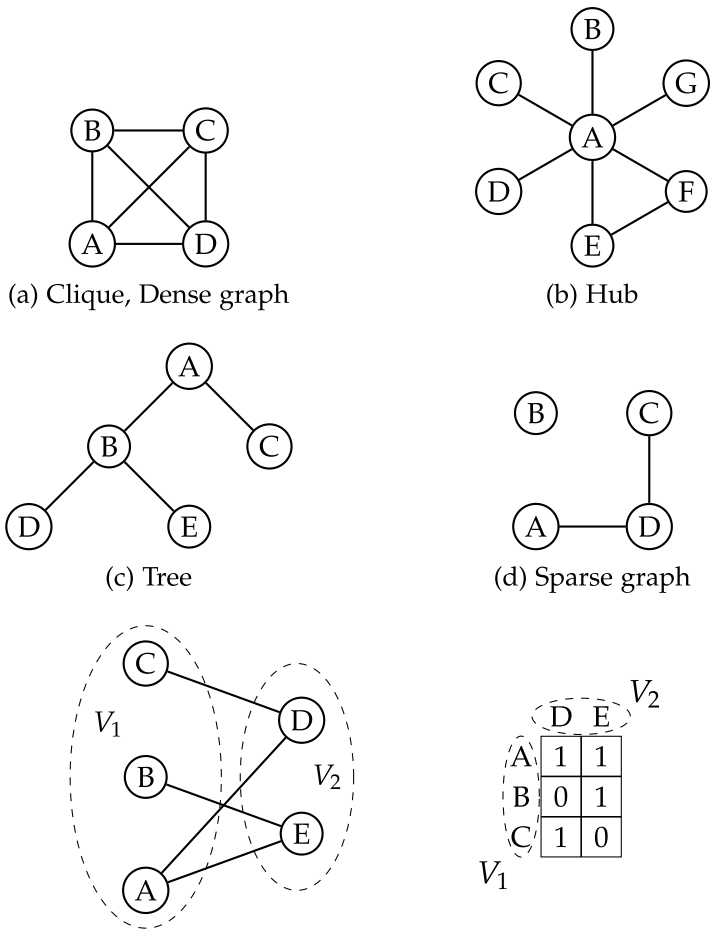
To understand information-theoretic clustering the following concepts are essential. All following random variables are discrete.

Entropy The entropy of a random variable X measures its level of information, thus, describing its uncertainty. It is defined as

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{2.3}$$

with the probability $p(x_i)$ for the outcomes of X.

A flip with a fair coin has two outcomes with equal probability $p(x_{head}) = \frac{1}{2}$ and $p(x_{tail}) = \frac{1}{2}$ and has the entropy $-(\frac{1}{2} \cdot \log_2 \frac{1}{2} + \frac{1}{2} \cdot \log_2 \frac{1}{2}) = 1$. Using the logarithm with base 2, the unit for entropy is called bit (binary digit). 1 bit is the maximal possible entropy of a random variable and expresses maximal uncertainty. An unfair coin has a lower entropy and contains less uncertainty. The definition of entropy is fundamental for the principle of *minimum description length* [Grü05] and other information-theoretic concepts like *minimum message length* [WB68]. It is closely related to the philosophical concept of Occam's



(a) Clique, Dense graph

(b) Hub

(c) Tree

(d) Sparse graph

(e) Bipartite graph with corresponding adjacency matrix

Figure 2.2.: Toy examples of various graph types (cf. Definitions 2.7 to 2.11).

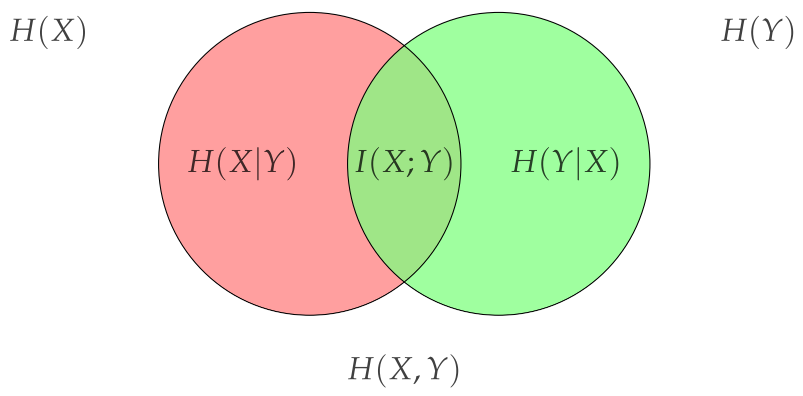


Figure 2.3.: Visualization of conditional entropy $H(X|Y)$, $H(Y|X)$ and Mutual Information $I(X;Y)$.

razor. This concept can be formulated as: *the simplest explanation has the highest probability to be correct.*

Conditional Entropy The conditional entropy measures the amount of uncertainty of a random variable X knowing the outcome of another random variable Y. If both random variables are independent, the conditional entropy equals the entropy as defined above. Conditional entropy is defined as

$$H(X|Y) = -\sum_{x \in X, y \in Y} p(x,y) \log \frac{p(x,y)}{p(y)} \tag{2.4}$$

with the joint probability $p(x,y)$ of two outcomes occurring together. If X and Y are independent, knowing the outcome of one random variable does not tell anything about the outcome of the other random variable; hence $H(X|Y) = H(X)$ and $H(Y|X) = H(Y)$. The relation between entropy and conditional entropy is visualized in Figure 2.3 by a Venn diagram. The figure also motivates the following concept.

Mutual Information Mutual information measures how much information the random variables X and Y share and how much the outcome of one random variable reduces the uncertainty of the other random variable. It is defined as

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x,y) \log \frac{p(x,y)}{p(x) \cdot p(y)}. \tag{2.5}$$

If $H(X|Y) = H(Y|X) = 0$, each random variable is completely determined by the other, and the mutual information equals the entropy of each of both random variables: $I(X;Y) = H(X) = H(Y)$. Figure 2.3 depicts mutual information as the intersection of the entropy of both random variables. If the random variables are independent ($p(x,y) = p(x) \cdot p(y)$), the mutual information is 0: they share no information. In information-theoretic clustering, an application for mutual information is to measure how much information cluster labels share with class labels, which is introduced below.

2.2.1. Minimum Description Length (MDL)

The clustering and partitioning algorithms proposed in this thesis (in Chapters 4, 7, and 8) base their objective function or clustering function, i.e., the internal measure to find the optimal assignment of objects to clusters, on the information-theoretic principle of the *minimum description length* (MDL) proposed by Rissanen [Ris78; Ris05; Grü05]. This concept aims to transform the clustering problem into a compression problem. The best clustering of a set of objects is the one that best compresses the objects and their attributes. Learning is understood as data compression. Regularities in a subset of objects help to reduce the length of a (hypothetical) code as the result of compressing the objects in a cluster. The more homogeneous the objects in a cluster are, the shorter is this code sequence.

The code length is also the subject of a more abstract (but not computable) concept: The *Kolmogorov complexity* of an object is defined as the length of the shortest program (in any general-purpose programming language) that produces this object. From a bird's eye perspective, MDL follows the principle of Occam's razor mentioned above.

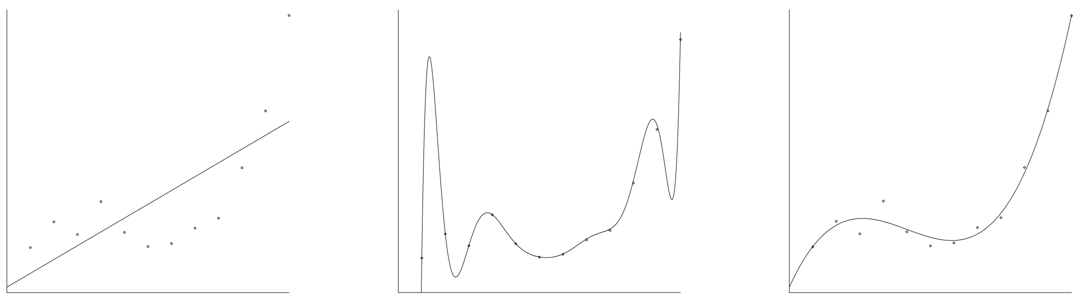


Figure 2.4.: Motivating the concept of minimum description length: Model complexity vs. goodness of fit (from [Grü05])

Crude MDL

The practical approach of *crude* MDL takes regularities in the data into account by creating a model or hypothesis that represents those regularities. The data D is then compressed by this model. The best model M compresses the data most without being overly complex. This results in a trade-off between model-complexity and data compression, thus, minimizing $L(M) + L(D|M)$, where $L(M)$ is the length (in bits) of the model, and $L(D|M)$ is the length (in bits) of the data compressed by the model. Minimizing both code length of model and compressed data avoids using an overly complex model and prevents overfitting.

This approach is motivated in Figure 2.4: twelve data points are fit by a most simple, linear function (left), a very complex 11-degree polynomial (middle), and a three-degree polynomial (right). Intuitively, the three-degree polynomial is a good trade-off between model complexity and the goodness of fit of the data to the model.

Refined MDL

While crude MDL is helpful for practice and is applied to the clustering algorithms in this thesis, it has its drawback: Each model has to be explicitly coded ad hoc, as, e.g., the functions are in Figure 2.4. Two different encodings for the same model may result in different code lengths. Thus, the model selection process is not entirely objective and can be biased. *Refined MDL* is an approach to avoid this by not encoding model and data separately in a two-part code as in crude MDL. Instead, a one-part code is chosen to encode the data in a way that relates to the *stochastic complexity* of the data, and implicitly the least complex model is preferred. Refined MDL gives a shorter code length the better the data fit the best-fitting model, eliminating the arbitrariness of models in crude MDL.

For the example in Figure 2.4, refined MDL uses the best-fitting third-degree polynomial to fit the data. In crude MDL, the parameters of the third-degree polynomial have to be explicitly selected and encoded separately. In practice, refined MDL is challenging to define, while the two-part codes of crude MDL are much easier to apply. Carefully choosing the models minimizes this drawback. Therefore, the complex models applied by the clustering algorithms in this thesis use the two-part code of crude MDL.

2.2.2. External Cluster-Validity Measures

External cluster-validity measures evaluate a clustering algorithm to an external, e.g., manually assigned *gold standard* or *ground truth*. They intend to reward correct groupings of objects while penalizing incorrect sets and missed splits of grouped objects. This thesis uses information-theoretic cluster-validity measures: the measure by Byron E. Dom (in the following referenced as Dom and applied in Chapter 8) and the widely-used measure *normalized mutual information* (NMI) (applied in Chapters 4 and 7). We will outline both methods and their differences.

Cluster-Validity Measure by Dom

In order to measure the quality of clustering algorithms, Byron E. Dom introduced an information-theoretic cluster-validity measure to quantify the similarity between two sets of labels [Dom01; Dom02]. The first set, K, contains a cluster label k_i for each object i. This mapping is the result of a clustering algorithm. The second set, C, resembles the ground truth and consists of a class label c_i for each object i. Set $\mathcal{K}$ contains the unique cluster labels of set K (one for each cluster), and set $\mathcal{C}$ contains the unique class labels of set C (one for each class). Dom's measure (we refer to it as $\text{Dom}_{C,K}$ or Dom) evaluates how well the cluster labels predict the class labels, i.e., it calculates the code length after encoding the class labels if the cluster labels are known.

If $|\mathcal{C}| = |\mathcal{K}|$, i.e., the number of classes and clusters match, $\text{Dom}_{C,K}$ is equivalent to the (estimated, i.e., observed) *empirical conditional entropy*

$$\widetilde{H}(C|K) = -\sum_{c=1}^{|\mathcal{C}|}\sum_{k=1}^{|\mathcal{K}|} \frac{h(c,k)}{n} \log \frac{h(c,k)}{h(k)} \tag{2.6}$$

which measures the information of the class labels given the cluster labels. The frequencies $h(c,k)$ and $h(k)$ are taken from the elements or columns, respectively, of the matrix or *contingency table* $\mathcal{H}$. An element h_{ck} represents the number of objects with class label c and cluster label k. If all objects of the same class are clustered together perfectly, the cluster labels fully describe the class labels; the conditional entropy equals 0. The conditional entropy is maximal if the cluster labels cannot describe the class labels (their mutual information equals 0).

If $|\mathcal{C}| \neq |\mathcal{K}|$, i.e, both partitions are of different sizes, the above definition prefers a larger to a smaller number of clusters. Following the principle of Occam's Razor, a more complex solution that adds little or no additional knowledge must be penalized. Using conditional entropy alone would always give one cluster per object as the best solution with zero entropy. With the information-theoretic idea that the information about the clusterings is encoded into bit sequences measured by the entropy defined above, sent over a channel, and decoded again, additional parameters are required beyond the encoded class labels. To be able to decode all the information, the decoder must know the frequencies $h(c,k)$. Thus, the contingency table $\mathbf{H}$ needs to be encoded without the distributions $h(k)$ of the objects in the clusters. Those we assume to be known to the encoder and decoder. This gives Dom's measure as the entire coding costs for data and model,

$$\text{Dom}_{C,K} = \widetilde{H}(C|K) + \frac{1}{n}\sum_{k=1}^{|\mathfrak{K}|} \log \binom{h(k) + |C| - 1}{|C| - 1} \tag{2.7}$$

Dom scores 0 for the *optimal* clustering: the cluster labels match the class labels (apart from renaming). The higher Dom scores, the worse the clustering is.

Normalized Mutual Information

Among the most widely used cluster-validity measures is the measure of *normalized mutual information* (NMI) [Yao03]. The mutual information of class labels C and cluster labels K is normalized to the lower bound 0 (worst possible clustering) and upper bound 1 (optimal clustering). Unlike the cluster-validity measure Dom (defined in Section 2.2.2) based on conditional entropy, NMI uses the uncertainty coefficient

$$U(C|K) = \frac{I(C;K)}{H(K)} \tag{2.8}$$

measuring the fraction of uncertainty in the class labels C given the cluster labels K. The Venn diagram in Figure 2.3 helps understand the relation between the uncertainty coefficient and Dom. The (symmetric) measure of normalized mutual information uses the harmonic mean[2] (for the average of rates) of the uncertainty coefficients $U(C|K)$ and $U(K|C)$, so that

$$\mathrm{NMI}_{C,K} = \frac{2}{\frac{1}{U(C|K)} + \frac{1}{U(K|C)}} = 2\frac{I(C;K)}{H(C)+H(K)}. \tag{2.9}$$

It scales from 0 if cluster labels give no information about class labels (and vice versa) to 1 if cluster labels and class labels are identical (apart from renaming).

Another popular cluster-validity measure (which is not used in this thesis but shall be mentioned) is the *adjusted mutual information* which aims to adjust the mutual information for *randomness*: a random distribution for cluster labels and class labels should result in the lowest value of 0.

2.3. Notations

The previous sections of this chapter have introduced our notations for graphs (Section 2.1) and information-theoretic clustering (Section 2.2). We summarize notations and definitions relevant to understanding a specific chapter at the beginning of each chapter. This section defines notations used throughout this thesis (cf. Table 2.1).

[2]There exist other approaches to normalize the mutual information. These use, e.g., the geometric mean of the uncertainty coefficients (cf. [VEB09]).

Table 2.1.: Important Symbols and Definitions.

Symbols	**Definitions**
$k, n, \mu, \ldots$	scalars
$\mathbf{u}, \boldsymbol{\mu}, \ldots$	column vectors
$\mathbf{u}^\mathsf{T}, \mathbf{v}^\mathsf{T}, \ldots$	row vectors
$\mathbf{S}, \boldsymbol{\Sigma}, \ldots$	matrices
	Clustering and Entropy
$n \in \mathbb{N}$	number of objects
$k \in \mathbb{N}$	number of clusters
$d \in \mathbb{N}$	dimensionality or the original space
$m \in \mathbb{N}$	dimensionality of the clustered subspace
X, Y	random variables
$H(X)$	entropy of random variable X
$H(X\|Y)$	conditional entropy of random variables X and Y
$I(X;Y)$	mutual information of random variables X and Y
	Graphs
V (V_i)	node-set of G (of subgraph structure G_i)
E (E_i)	edge-set of G (of subgraph structure G_i)
$G = (V, E)$	(directed or undirected) unweighted graph
$G = (V, E, w)$	(directed or undirected) (edge-)weighted graph
$w : E \mapsto \mathbb{R}$	(edge-)weight function
G_i	i-th subgraph structure
A	adjacency matrix of G
n (n_i)	number of nodes of G (of subgraph structure G_i)
m (m_i)	number of edges of G (of subgraph structure G_i)

2.3.1. Scalars, Vectors, and Matrices

To denote **scalars**, we use lower-case Roman and Greek letters; e.g., the number of clusters $k \in \mathbb{N}$, the number of objects $n \in \mathbb{N}$, or the mean $\mu \in \mathbb{R}$ of a probability distribution. **Column vectors** are denoted by boldface lower-case Roman ($\mathbf{u}$) and Greek letters (e.g., $\boldsymbol{\mu}$ as the center of a multidimensional cluster). **Row vectors** are denoted as transposed column vectors ($\mathbf{v}^\mathsf{T}$). **Matrices** are indicated by boldface upper-case Roman ($\mathbf{S}$) or Greek letters (e.g., the covariance matrix $\boldsymbol{\Sigma}$). Boldness can be omitted if the context allows no ambiguity.

2.3.2. Clustering and Entropy

We denote the **number of objects** to cluster with n and the **number of clusters** with k. In subspace clustering, d represents the **dimensionality of the original space**, and m the **dimensionality of the clustered subspace**. Our proposed clustering algorithms use the minimum description length (MDL) to avoid input parameters. MDL is based on the concepts of **entropy** $H(X)$ of a **random variable** X (cf. Section 2.2). For the random variables X and Y, the **conditional entropy** is denoted by $H(X|Y)$, and the **mutual information** that both random variables share by $I(X;Y)$.

2.3.3. Graphs

Summarizing our notations introduced in Section 2.1, we denote a (directed or undirected) **unweighted graph** with $G = (V, E)$. V represents the set of **vertices** or **nodes**, and E the set of **edges**. A **weighted graph** is denoted by $G = (V, E, w)$ with an (edge-)weight function $w : E \mapsto \mathbb{R}$. The **number of nodes** in a graph G is expressed by n, and the **number of edges** in a graph G by m. Analogously, $G_i = (V_i, E_i)$ represents a **subgraph** or **subgraph structure** i with its set of nodes and its set of edges. With n_i and m_i, we denote the number of nodes and edges in a subgraph structure i. The number of identified subgraph structures in a graph is defined as k (analogously to the notation for the number of clusters commonly used in clustering). The **adjacency matrix** of a graph G is marked as A and the **degree** of a node v of graph G as $d_G(v)$.

2.4. Reproducibility

We strongly believe that all our experiments proving the value of our proposed methods must be reproducible by other researchers as good scientific practice demands. Furthermore, practitioners can directly apply all our solutions and algorithms to benefit from the results of this thesis. Therefore, we make all our implementations publicly accessible. For the experiments in our evaluations, we either use synthetic or real-world data that are already publicly accessible and disclose our sources when data is used, or we provide the data as well. The implementations and data are available at `https://s.goebl.net/research`.

Part II.

Fast and Effective Methods for Explainable Graph Structuring and Summarization

Graphs are a valuable instrument to abstract complex problems involving any form of connectivity or interaction. Questions like "how to detect fraudulent activity in online trust networks?" or "how to find regions of interacting activity in the human brain caused by a stimulus?" are much easier to grasp when mapped to mathematically well-defined graphs. It opens up a rich toolbox of analytical methods to expand knowledge about the data. Graph theory has been subject to research at least since Leonard Euler and the Seven Bridges of Königsberg nearly 200 years ago. In this part, we contribute to this toolbox by introducing two methods that explain the inner structure of graphs. We break them down into meaningful partitions and reveal unknown patterns or suspicious behavior.

Our first method *MeGS* (Partitioning **Me**anin**g**ful **S**ubgraph Structures using Minimum Description Length) structures graphs into meaningful components by analyzing the edges of the graphs. It places each node into an interpretable context that helps the user to extract knowledge from the data. No unknown areas remain in the graph. Thanks to MDL, MeGS is parameter-free and automatically finds the best number of components in a graph. Finding patterns in a graph is also the goal of algorithms for compressing graphs. Hence, we touch on this problem with MeGS as well.

With our second method *Spectral Lens*, we propose an efficient tool that analyzes even larger graphs with millions of edges in minutes and explains the connectivity patterns of subsets of nodes in the graph. By highlighting the connectivity of outlier nodes, Spectral Lens identifies fraudulent activity in contrast to areas of similar connectivity. It covers the full range of unweighted and weighted graphs as well as undirected and directed graphs.

The author's previously published work incorporated in Part II This part of the thesis with our proposed algorithms MeGS and Spectral Lens is based on and has been partly published in the following two publications that have been accepted and published in peer-reviewed proceedings of the IEEE International Conference on Data Mining [Goe+16; GKF17]:

1. S. Goebl, A. Tonch, C. Böhm, and C. Plant. MeGS: Partitioning Meaningful Subgraph Structures using Minimum Description Length. In: *ICDM 16:*

Proceedings of the 2016 IEEE 16th International Conference on Data Mining. 2016, pp. 889-894.

2. S. Goebl, S. Kumar, and C. Faloutsos. Spectral Lens: Explainable Diagnostics, Tools and Discoveries in Directed, Weighted Graphs. In: *ICDM 17: Proceedings of the 2017 IEEE 17th International Conference on Data Mining.* 2017, pp. 877-882.

In the following chapters, we present our algorithms MeGS and Spectral Lens in extended versions with about 60% more content than published in papers 1 and 2. This extra content has been peer-reviewed and approved in the original submissions but was cut out due to space limitations (short papers).

Both papers were mainly composed, presented, and managed by the first author. Claudia Plant (University of Vienna, Austria), Christian Böhm (University of Munich, Germany), and Christos Faloutsos (Carnegie Mellon University, PA, USA) helped with supervision and mentoring and accompanied the process of the creation of the papers. Annika Tonch (University of Munich, Germany) provided the BRAIN dataset (Section 4.4.2) and supported the project with regular discussions. Srijan Kumar (Georgia Institute of Technology, GA, USA) helped with discussions, comments, and suggestions throughout the project and provided the BITCOIN-OTC dataset (Section 5.4.2).

Structure of Part II Chapter 3 gives an overview of research related to our proposed methods. Chapter 4 presents our first graph mining algorithm MeGS for partitioning graphs into meaningful subgraph structures. Chapter 5 introduces our proposed method Spectral Lens for explainable diagnostics of large directed and undirected, and weighted and unweighted graphs.

3. Related Work

Past research related to our proposed methods MeGS and Spectral Lens forms four groups: *graph structuring and summarization*, *graph clustering and partitioning*, *graph compression*, and the search for *regularities and suspicious behavior in graphs*. We give a short survey on the related work with a section covering each group.

3.1. Graph Summarization, Structuring, and Compression

A key challenge to data mining lies in simplifying complex data for human understanding. Massive graphs with millions of nodes and edges are hard to grasp, impossible to visualize with all details, and difficult to further process efficiently in domain-specific applications. Methods for graph summarization, structuring, and compression meet this challenge and break large networks down into smaller summary graphs, and structure nodes into meaningful (overlapping or non-overlapping) synoptic groupings of nodes and edges.

The recent comprehensive survey by Liu et al. [Liu+18] divides the approaches for graph summarization (include structuring and compression) by its core techniques into four areas:

- *Grouping and aggregation-based approaches.* Node- or edge-groupings aggregate nodes or edges into meaningful groupings that describe the substructures of a graph. This approach is also taken by our proposed method MeGS which groups nodes into meaningful non-overlapping structures.

- *Bit compression-based approaches.* Lossless and lossy graph compression optimizes graphs for efficient further processing. The information-theoretic approach that postulates that the optimal compression is gained by graph primitives describing the graph best is shared by our proposed method MeGS.

- *Simplification or sparsification-based approaches.* The simplified graph contains a subset of nodes, leaving out what is assumed to be the least important.
- *Influence-based approaches.* A graph is described by its patterns of influence-propagation on a high level.

Our proposed method MeGS combines approaches of the first two areas and facilitates graph summarization, structuring, and compression. Related work in these areas is discussed in the following under the aspects of graph summarization and structuring (Section 3.1.1), and graph compression (Section 3.1.2).

3.1.1. Graph Summarization and Structuring

Several MDL-based algorithms exploit the inner structure of graphs (mostly) without requiring any input parameters.

CXPrime [Fen+13] discovers patterns in sparse graphs. CXPrime exploits structure primitives using the MDL principle so that the number of clusters is selected automatically. However, CXPrime only detects cliques and hubs due to its restriction to three-node primitives. Another limitation of the method is that it searches for the optimal number of clusters in an interval that requires an upper limit as input.

VoG [Kou+14] finds a set of possibly overlapping subgraphs using the MDL principle. Due to the overlap, it does not fully partition a graph as our proposed method MeGS does. Furthermore, VoG relies on other algorithms for decomposing the graph.

Subdue [KHC05] discovers subgraph structures that are frequent and that best compress the graph dataset according to the MDL principle. However, Subdue often discovers only very tiny subgraph structures, sometimes only one-edge primitives. The frequent structures also tend to be heterogeneous. Both characteristics barely give the user interpretable subgraph structures, and it is not suitable to help understand a graph. On top of that, long-running time makes it hard to process large datasets.

Among other algorithms for the discovery of frequent subgraph structure are gSpan [YH02], FSG [KK04], and GBI [Mat+00], to mention a few.

Navlakha et al. [NRS08] propose a two-part graph representation that adds a set of corrections to a graph summary. The graph summary consists of nodes representing sets of nodes and edges representing edges between all pairs of nodes in the connected sets of nodes. Their proposed algorithm creates lossless summaries as well as lossly summaries with a bounded error. Using the MDL principle, the authors aim to find the best summary along with a set of edge corrections.

In [Fen+12], the summaries restrict to identifying bipartite sets using MDL. Like Navlakha et al., it constructs much smaller graph representations to support the interpretation of the original graph. However, unlike node- or edge-groupings, a summary graph gives no explicit qualitative information about the graph or groups of its nodes.

Matrix Decomposition [Mie15; MV14] takes a different MDL-based approach to extract structural information: Overlapping structures are identified by factorizing the adjacency matrix into sub-matrices.

Graph summarization and structuring methods facilitate the visualization of the adjacency matrix of a graph by reordering nodes and labeling grouping with structural information like our proposed method MeGS as well as, e.g., Cross-Associations or SlashBurn. Other approaches to visualize graphs and their structures include plotting the distribution of connected components even for massive, petabyte-scale graphs (*PeGaSus* [KTF11])), or visualizing outliers in graphs (*Oddball* [AMF10]; cf. Section 3.3.2).

3.1.2. Graph Compression

Due to the increasing size of graphs, many algorithms deal with the efficient compression of data, like [BV04] and [Chi+09]. Graph compression is relevant for storing and transferring large networks and also allows to keep large graphs in memory for efficient further calculations. Many information-theoretic graph summarization methods that use the minimum description length principle produce the optimal grouping or structuring of nodes by optimizing the graph compression. So does, e.g., *Cross-Associations* compress the graph in the process of finding a clustering.

SlashBurn [KF11] also uses the information-theoretic approach to compress the data using specific primitives. It compresses graphs by looking for hubs and spokes and recursively envisions graphs as a collection of hubs connecting hubs, with super hubs connecting the hubs, and so on. However, the compression rate of Slashburn depends on the input parameter block width.

VoG and *CXPrime* also compress graphs using MDL. With our proposed method MeGS, graph compression comes as a side-effect but is not our primary goal.

Navlakha et al. [NRS08] not only create graph summaries using lossless compression but also reduce the size of the summary graph further with lossly compression. To preserve the quality of the summary graph, a bounded error is given.

For further related work, see the survey by Liu et al. [Liu+18].

3.2. Graph Clustering and Partitioning

Graph clustering and graph partitioning are well-established topics in research. Traditionally, a cluster is a grouping of nodes maximizing their similarity to each other while minimizing the similarity between clusters. Many approaches deal with several topics of graph clustering, like *Markov-Clustering* [Don08]. For a survey, see [AR13].

Graph partitioning follows the same goal of grouping nodes but focuses on separating subsets of nodes.

Metis [KK99] is a multilevel k-way hypergraph partitioning algorithm that has attracted much attention. However, it requires the number of partitions as an input parameter.

The minimum description length principle helps algorithms become parameter-free: besides [Fen+12], *Cross-Associations* [Cha+04] cluster large sparse binary matrices. Due to MDL, it is fully automatic and simultaneously discovers row and column groups.

With graph clustering among its applications, *kernel functions on graphs* [Fou+12] have been proposed as similarity measures between nodes of graphs. Comparing the nodes allows us to categorize them into structures as well. The related topic of *graph kernels* [Bor+20] applies a kernel function on the entire graph, e.g., for transforming it into input to a machine learning algorithm without needing a separate feature-extraction step. For given subgraphs, this can also be used as a method to categorize by similarity.

The work in graph clustering and partitioning is related to our proposed method MeGS. However, MeGS goes beyond simple clustering and finds various groups with different characteristics. To enhance this difference, we call our groups structures, not clusters.

3.3. Regularities and Suspicious Behavior in Graphs

Detecting regular and irregular behavior in graphs is a different approach to investigating graphs and networks. It overlaps with the strategy of finding structures and clusters, which one can see as regular patterns. This topic is related to our proposed method Spectral Lens that can process directed and undirected and weighted and unweighted graphs. Therefore, we look at related work from this point of view.

3.3.1. Finding Regularities in Graphs

Finding regularities in graphs and partitioning these regularities into subgraphs is a prevalent task. Some approaches define regularities as cliques or bipartite subgraphs (cf. *Cross-Associations* [Cha+04]) or as other primitives like hubs (cf. *CXPrime* [Fen+13]). Cross-Associations and CXPrime can only process unweighted graphs.

Co-clustering [DMM03] clusters nodes of a graph by maximizing mutual information and requires the number of clusters as an input parameter.

Other methods use cut-based multilevel approaches to detect communities like *Metis* [KK99] and *GraClus* [DGK07] or use singular value decomposition like [Dri+; SD11].

Also, *kernel functions* are applied to graphs to obtain similarities between nodes [Fou+12].

Spectral clustering (survey: [DGK07]) also is cut-based and finds clusters in a graph.

Graph summarization techniques (cf. Section 3.1.1) use regularities to summarize or compress the graph: *VoG* [Kou+14] finds primitives like hubs and cliques for compression by minimum description length. *Subdue* [KHC05] discovers substructures in a graph and also uses them for compression.

Eigenspokes [Pra+10] finds communities in a graph by decomposing the adjacency matrix into singular vectors. However, Eigenspokes can not process edge weights.

The concept of *bridges* that we introduce with Spectral Lens has not yet been discussed in the related work. The closest but not identical concept is *overlapping communities*: an overlap of three communities (A, B, C) does imply there is a connection between all of these communities; a *partial* similarity, i.e., between A and B as well as between A and C but not between B and C can not be expressed. Overlapping communities have been approached, e.g., by matrix factorization (*BigClam*[YL13]), probabilistic model-fitting (*AGM*[YL12]) or label propagation (*COPRA*[Gre10]); unlike Spectral Lens, none of these methods can process negative edge-weights.

3.3.2. Detecting Suspicious Behavior in Graphs

Several methods have been proposed for highlighting suspicious behavior in graphs.

Oddball [AMF10] looks for anomalous nodes in weighted graphs. It defines rules for a "neighborhood sub-graph" and detects deviations from these rules as anomalies.

FBOX [Sha+14] finds small-scale attacks in networks that spectral methods do not detect.

Decluttering [KSS14] identifies suspicious nodes by their local edge connectivity with respect to the entire network.

Suspicious behavior in graphs also occurs when nodes behave in unusual synchronous patterns, a so-called *lockstep behavior*. *FRAUDAR*[Hoo+16], *LOCK-*

STEP[Jia+14b], *EdgeCentric*[Sha+16], *CopyCatch* [Beu+13] and *CatchSync* [Jia+14a] detect nodes that are part of these patterns which may be due to hijacked or scripted accounts.

Unlike our proposed method Spectral Lens, none of the current work identifies and explains both the groups of regular and irregular behavior as well as groups with similar edge-connectivity patterns in directed and (positive or negative-) weighted graphs. Also, no comparison method automatically finds interpretable partitions that include highlighting bridges and overlaps.

4. MEGS: Partitioning Meaningful Subgraph Structures using Minimum Description Length

How can we fully structure a graph into pieces of meaningful information? Into structures that provide us with insights and carry a meaning beyond simple clustering? How can we also exploit these patterns to compress the graph for fast transmission and easier storage? —

In many applications of graph analysis like network analysis or medical information extraction we are searching for special patterns. Here, it is not sufficient to extract only parts of the relevant information in a graph, but to understand the complete underlying structure. Therefore, we propose our algorithm MeGS (Partitioning Meaningful Subgraph Structures using Minimum Description Length) to fully understand how a graph is constructed. The most common primitives (clique, hub, tree, bipartite, and sparse) serve as models to split a graph into meaningful structures. Using the principle of minimum description length (MDL) structure types and counts are determined by the best fitting model. These structures achieve the best compression of the adjacency matrix. As result, every node is part of exactly one structure and has an interpretable context. No unknown areas remain in the graph. The higher a model compresses its section of the graph, the stronger its match with the corresponding structural assumption. MeGS, a fast and parameter-free split-and-merge algorithm, automatically finds the optimal structures achieving the best compression. We compare to state-of-the-art algorithms to prove MeGS' ability for interpretation and compression.

4.1. Introduction

How can a graph and its subgraph structures be fully understood? We propose that a complex graph can be represented with a succinct set of patterns which are readily understood by practitioners from the problem domain. Many applications in network analysis or medical information extraction require a full understanding of how and by what parts a graph is constructed. If a graph corresponds in all its parts to patterns well known to the user, he or she finds his or her ideas well represented. Hence we say these patterns carry meaning for the user and are *meaningful structures*. We aim to find a structural representation containing all nodes of a graph, not only some subsets. For clear understanding and easy interpretation we assign each node to exactly one structure. We provide an expandable dictionary of the most common primitives (clique, hub, tree, bipartite, and sparse). We identify the structures and visualize the graph by permuting its adjacency matrix. The *minimum description length* (MDL) principle (cf. Section 2.2.1) guides the partitioning in number and types of the subgraph structures.

Various algorithms have been proposed to partition graphs, as has been summarized in the literature review in Chapter 3. We recapitulate the closest competitor algorithms to MEGS (c.f. Table 4.1). Metis [KK99] finds a given number of communities (or cliques). This simple paradigm clusters nodes together if they show many edges in-between, and separates them if they are loosely connected. Other methods use the information-theoretic MDL to fit node and edge sets to models so that the number of partitions is automatically defined: Cross-associations (CA) [Cha+04] permutes the adjacency matrix in order to find dense partitions (cliques and bipartite structures). However, the complexity of different graph structures has not yet been fully covered.

Another approach is given by SLASHBURN [KF11] which is based on a hub-like paradigm and helps to visualize a graph. Being MDL-based as well, it decomposes a given graph by removing the highest-degree nodes step-by-step. VoG [Kou+14] and CXPrime [Fen+13] have made approaches towards the identification of graph patterns using MDL. VoG does not include the complete graph in its structure detection. Thus, it only allows interpretation of parts of it, but not of the full graph. VoG also detects overlapping structures. While in some

Table 4.1.: Comparison of MeGS and related algorithms in various goals of graph analysis which MeGS contributes to.

	Structuring	> 2 Primitives	Partitioning	Visualization	MDL-based
MeGS	✔	✔	✔	✔	✔
CXPrime [Fen+13]	✔		✔	✔	✔
VoG [Kou+14]	✔	✔			✔
CA [Cha+04]			✔	✔	✔
Metis [KK99]			✔		
SlashBurn [KF11]				✔	✔

applications overlapping structures may be asked for, we will show the value of an explicit node assignment in our experiments: this allows a clear visualization and interpretation of all structures. Therefore, we do not include overlapping structures. Like our method, CXPrime gives a clear structure assignment based on MDL. However, CXPrime has a very limited and non-expandable dictionary (clique and hub) and is not parameter-free.

Our proposed algorithm MeGS (Partitioning Meaningful Subgraph Structures using Minimum Description Length) gives a full understanding of the underlying structures of a graph. Figure 4.1 gives a toy example: the graph contains five prevalent structures: a clique-like, a hub-like, a bipartite-like, a tree-like structure, and a cyclic, planar structure.[1] None of the structures manifests an ideal pattern, but all contain errors in terms of missing or additional edges. E.g., the tree-like structure shows cyclic edges; the spoke nodes of the hub are partly connected; the bipartite sets show edges not only between but also in each set. In addition to identifying the best-fitting structures, the algorithmic task is to separate the structures which are all connected to each other (partitioning task).

[1]The planar structure is characterized by too many cyclic edges to be regarded as tree-like, but is too sparse to appear clique-like.

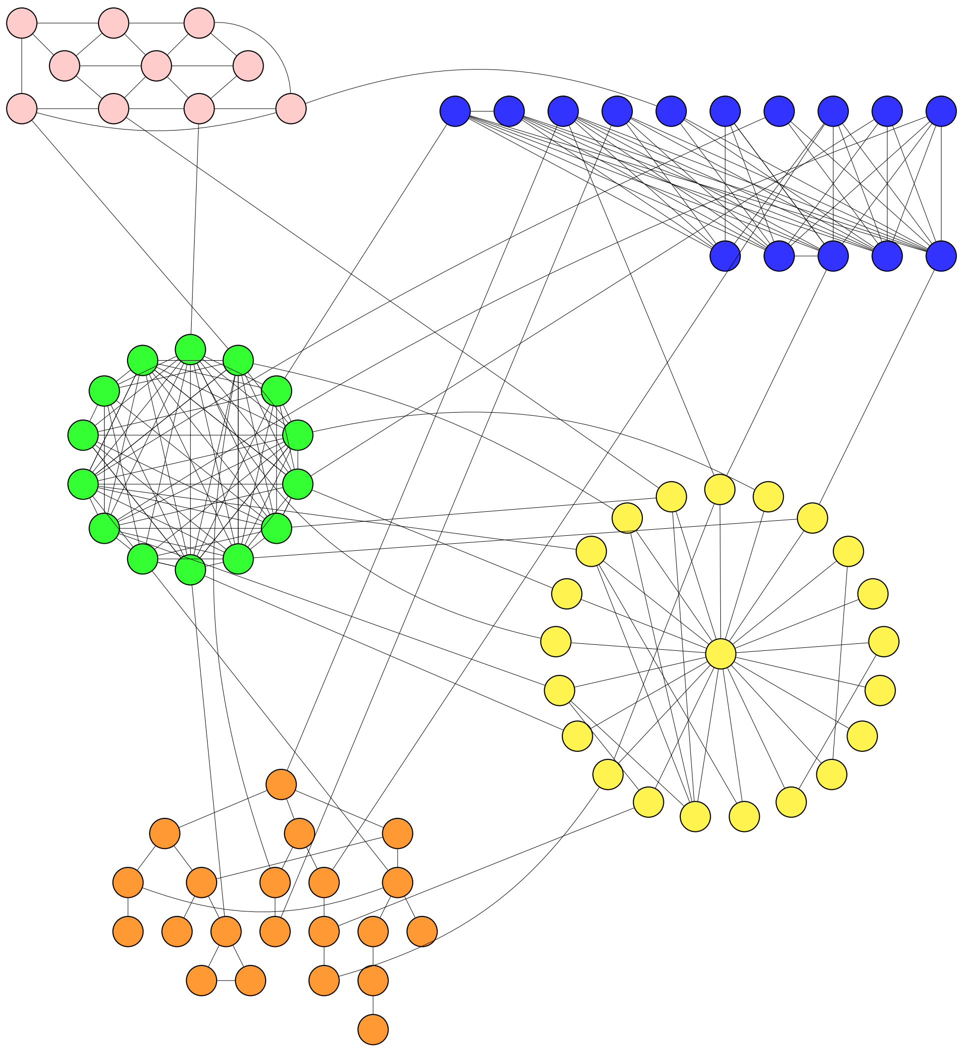


Figure 4.1.: Synthetic graph illustrating the idea of MeGS. The nodes are color coded by the corresponding structures (Clique 🟢, Bipartite 🔵, Tree 🟠, Hub 🟡 and Sparse ●).

We summarize our **contributions**:

1. We present MeGS, an algorithm that simultaneously *partitions* a graph into a range of *meaningful structures* and *enhances visualization* by *permuting* its adjacency matrix (features not offered by existing algorithms, c.f. Table 4.1). Every node is part of exactly one structure and has an interpretable context.

2. By using MDL, MeGS is *parameter-free* in theory and practice, and *losslessly* compresses the graph.

3. Extensive experiments demonstrate that MeGS can be used to *reveal structural information* in real-world graphs.

To the best of our knowledge there is no other proposed method with these contributions that is able to extract all our given structures (clique, hub, bipartite structure, tree and sparse structure) and, thus, to provide a distinct node assignment and interpretable visualization which helps understand a graph and its inner structure. The above mentioned state-of-the-art methods will be compared in detail in our experiments.

The rest of this chapter is structured as follows: In Section 4.2 we present the theory behind MeGS and build the underlying MDL compression schema. The algorithm MeGS is given in Section 4.3. Section 4.4 contains experiments on synthetic and real-world data. Section 4.5 concludes the chapter.

4.2. Meaningful Graph Structures

To identify meaningful structures we define a detailed MDL compression schema. The overall compression length defined by MDL is our objective function that guides our algorithm. It is minimized if the optimal number of structures and the best fitting types of structures are selected. Thus, a sound objective function is vital for success. To ease comprehension Table 4.2 gathers important definitions.

We follow the information-theoretic principle of Shannon's entropy (cf. Section 2.2): if we have no a-priori knowledge about an arbitrary graph, the amount of information and, thus, the bit length for compressing the graph is defined as follows: we regard the elements 1 (edge) or 0 (no edge) of the $n \times n$ adjacency

Table 4.2.: Important Symbols and Definitions.

Symbol	**Definition**
G (G_i)	graph (i-th subgraph structure)
A	adjacency matrix of G
V (V_i)	node-set of G (of subgraph structure G_i)
n (n_i)	number of nodes of G (of subgraph structure G_i)
E (E_i)	edge-set of G (of subgraph structure G_i)
E_i^0	edge-set of subgraph structure G_i as fully-connected subgraph
$E_\times$ ($E_\times^0$)	edge-set between all structures (analogously as fully-connected)
E_i^H; E_i^S	edge-set between hub node and spoke nodes; between spoke nodes
E_i^{A-B} (E_i^A, E_i^B)	edge-set between (within) bipartite sets A and B
E_i^C	cyclic edge-set (i.e. edges closing cycles)
k	number of subgraph structures
$\mathfrak{T}$	code book for structure compression
$t_i \in \mathfrak{T}$	type of structure i (e.g. *tree*)
C_i (PC_i, CC_i)	compression cost for structure i [bit] (model cost, edge cost resp.)
mdl_G	total encoding length of G [bit] (=objective function of MeGS)

matrix A of an unweighted graph G as a sequence of observations from an i.i.d. random variable with the distribution P_A (we follow [Cha+04]). In the adjacency matrix A the random variable produces either an edge or no edge, hence $P_A(0) = 1 - P_A(1)$. Now we can use the binary Shannon entropy function to measure the information contained in our random variable and, thus, its compression in bits as $H(P_A)$. The complete adjacency matrix is compressed by $n^2 \cdot H(P_A(1))$ bits, since $H(P_A(1)) = H(P_A(0))$. With V representing a set of nodes and E a set of edges of G, we can equally express the MDL of the adjacency matrix A as $|V|^2 \cdot H(P_A(1))$. Since $P_A(1) = |E|/|V|^2$, the compression length is equivalently expressed as $|V|^2 \cdot H(|E|/|V|^2)$ (as in [Cha+04]). This is the total bit length of a graph $G = (V, E)$ after compression. Since we restrict to undirected graphs without self-loops, the actual cost is lower. However, until now, we still use no structural information and, therefore, have bad average compression rates.

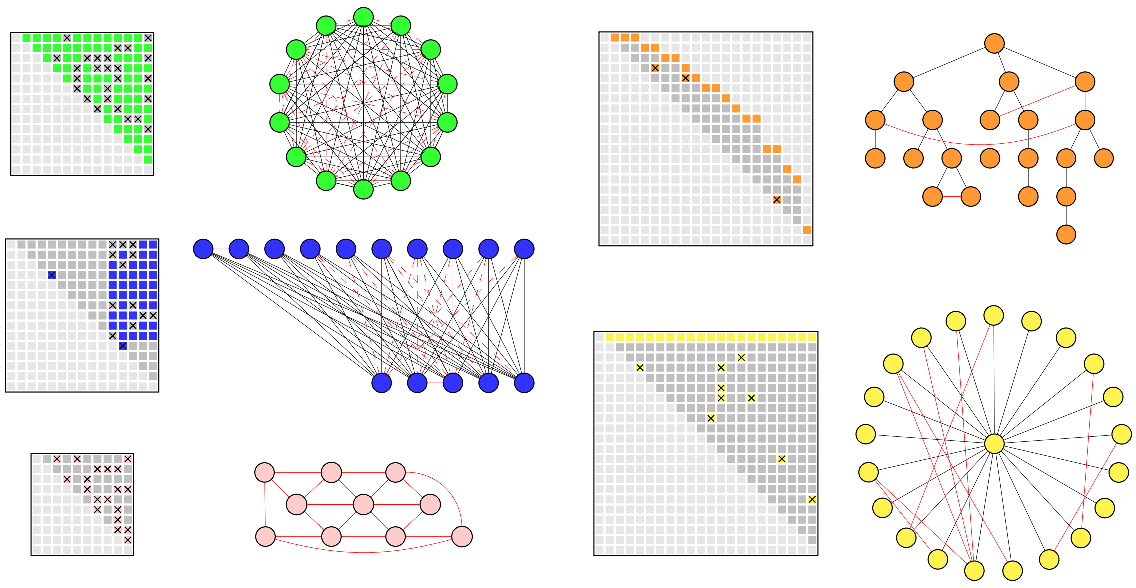


Figure 4.2.: Illustration for MDL structure encoding schema for structures Bipartite ●, Tree ●, Clique ●, Hub ● and Sparse ● from Figure 4.1. Deviations from the ideal structure by missing edges (- - -) or additional edges (—) are marked by an × in the adjacency matrix and require additional encoding by MDL.

As in information-theoretic data mining, we do not immediately compress the edges, but try to better describe the graph information by a-priori models. Then, only the model parameters and the deviation from the models by missing or additional edges have to be compressed. We gain structural information and better compression rates. After transmitting the data, lossless decompression only requires knowledge about the a-priori models.

We use meaningful graph structures as models for the following reasons: (1) They are common among real data graphs and allow high compression rates. (2) Already the information about type and number of structures in a graph delivers important knowledge about it. (3) Finally, the compression quality allows comparing to other algorithms that also use MDL and gives insights about how well models fit.

4.2.1. Graph Structure Encoding

Our motivation for using graph structures is their pervasiveness, simplicity and *meaningfulness*, i.e. they contribute to the interpretation of a graph. A pattern is not suitable if it only improves compression but not understanding of a graph. We propose a code book (or dictionary) consisting of following structures: cliques, hubs, bipartite structures, trees and sparse structures. However new structures accompanied by a (de-)compression algorithm can easily be added. The example in Figure 4.1 shows all five structures. With reference to it we now elaborate the encoding schema for the graph structures of our code book. Edges between structures will be discussed later.

To enhance understanding Figure 4.2 presents each structure of Figure 4.1 separately in node-edge as well as in adjacency matrix representation. The latter is relevant for compression. Since we restrict to undirected graphs without self-loops, the lower triangle of the adjacency matrix can be ignored: light gray don't-care edges (▫). Matched to a structural model, the graph can contain missing and/or additional edges. These edges resemble false negatives and false positives, whereas dark gray edges (▪) represent true negatives. Applied to a structure, the compression cost C_i for each structure i is composed of model parameter information PC_i and coding cost of edge information CC_i ($C_i = PC_i + CC_i$). All models include as parameter cost PC_i the type of the substructure as listed in the code book $\mathfrak{T}$ of $\lceil \log_2 |\mathfrak{T}| \rceil$ bits (= 3 bits for our code book), the size of each structure by $\lceil \log_2 |V| \rceil$ bits and for all member nodes V_i its structure ID using $|V_i| \cdot \log_2(|V|/|V_i|)$ bits. For defining CC_i each structure follows a different encoding schema:

Clique (●)

In an ideal clique i each node is connected to all other nodes. The matrix elements above the diagonal contain all edges or missing edges and are encoded using entropy. Thus, there are $|E_i^0| = (|V_i|^2 - |V_i|)/2$ possible edges (= fully-connected, symmetric and without self-loops). With $|E_i|$ being the number of existing edges above the upper diagonal $CC_i = |E_i^0| \cdot H(|E_i|/|E_i^0|)$ bits. Following fundamental insights of information theory [Ris07] adding the edge probability demands

$\frac{1}{2} \cdot \log_2 |V_i|$ bits as further parameter costs. Since a clique is characterized by a high number of edges, we define the condition $|E_i| >= |E_i^0|/2$.

Hub (●)

A hub shows a distinguished node that is connected to all other nodes of this structure, so that the hub node is of highest degree in the structure. Therefore, every missing edge from this node to all other nodes of a hub needs to be encoded as well as all additional connections between the remaining nodes. The edges E_i^H to the hub node require $(|V_i| - 1) \cdot H(E_i^H/(|V_i| - 1))$ bits. All other edges E_i^S between spokes are encoded by $(|E_i^0| - |E_i^H|) \cdot H(E_i^S/(|E_i^0| - |E_i^H|))$ bits. To differ from tree and clique, a hub requires at least half of the possible connections from the hub node to all spoke nodes ($|E_i^H| \geq (|V_i| - 1)/2$) and not more than half of the possible connections between the spoke nodes ($|E_i^S| \leq (|E_i^0| - |E_i^H|)/2$) As specific parameter cost the edge probabilities of hub and spokes have to be compressed in altogether $2 \cdot \frac{1}{2} \log_2 |V_i|$ bits.

Bipartite (●)

A perfect bipartite graph is divided into two disjoint set of nodes V_i^A, V_i^B so that all nodes of each set show edges E_i^{A-B} to all nodes of the other set. No inner edges E_i^A, E_i^B between nodes of a single set exist. Therefore, all edges E_i^{A-B} are arranged in a rectangle in the adjacency matrix. *Missing edges between* the bipartite sets and *additional edges in* the bipartite sets are deviations from the perfect bipartite model and are separately entropy encoded. Encoding edges between the bipartite sets is done by encoding the respective subset of the adjacency matrix: in the blue bipartite subgraph of Figure 4.2 all edges between $V_i^A = 1^{st}$ to 10^{th} node and $V_i^B = 11^{th}$ to 15^{th} node. Analogously to the full encoding of an adjacency matrix using $|V|^2 \cdot H(|E|/|V|^2)$ bits (derived above) existing edges and missing edges between the sets are encoded using $|V_i^A| \cdot |V_i^B| \cdot H(E_i^{A-B}/(|V_i^A| \cdot |V_i^B|))$ bits. Furthermore, there is a maximum of $E_i^{0^{A+B}} = E_i^0 - (|V_i^A| \cdot |V_i^B|)$ additional edges possible in both single sets. Thus, entropy encoding of existing edges and missing edges requires $E_i^{0^{A+B}} \cdot H((E_i^A + E_i^B)/E_i^{0^{A+B}})$ bits here. Both encodings compose CC_i. To be *meaningful*, bipartiteness requires, analogously to clique and hub, $E_i^{A-B} \geq \frac{1}{2} \cdot |V_i^A| \cdot |V_i^B|$ and $E_i^A + E_i^B \leq \frac{1}{2} \cdot (E_i^0 - (|V_i^A| \cdot |V_i^B|))$. Edge probabilities between and in bipartite sets are added to PC_i as $2 \cdot \frac{1}{2} \log_2 |V_i|$.

Tree (●)

A tree is defined as an acyclic bipartite connected graph. We arrange the nodes in breadth-first search order (BFS): nodes with the same parent node keep their original order in the adjacency matrix to reduce encoding cost. Then, the nodes on the next level follow in the order of their parent nodes on the higher level. Again, nodes with the same parent keep their original order. Thus, for every node except the root node we only need to store its parent node to encode the permutation, demanding altogether $(|V| - 1) \cdot \log_2(n_i - 1)$ bits. Since no information is given which node is the root node, and assumptions are difficult (balanced or unbalanced tree, max-degree root etc.), the first node of the adjacency matrix of the subgraph is chosen as root node.

The adjacency matrix of a perfect tree corresponds to the shape of descending stairs with only one edge per column if we consider only the upper triangular matrix (cf. Figure 4.2). Please note an important effect of BFS-ordering on the adjacency matrix: by definition no edges lie above the BFS-ordered tree edges. Hence, the these edges are marked as don't-care (■) and require no encoding. Edges which close cycles (cyclic edges) E_i^C represent deviations from the tree model and occur between the diagonal and below the tree edges in Figure 4.2. With $\text{children}_i^{v_m}$ being the set of non-cyclic edges of the m-th node of tree i, we sum up the number of all possible cyclic edges of tree i as

$$|E_i^{C_0}| = \sum_{v_j \in V_i} \left(\sum_{m<j} |\text{children}_i^{v_m}| - (j-1) \right). \tag{4.1}$$

These edges are both the dark gray (■) and crossed out (×) edges. Now, we can state the entropy cost for cyclic edges as $|E_i^{C_0}| \cdot H(E_i^C / |E_i^{C_0}|)$ bits. Note that a hub is a special case of a tree. To differ from a hub structure, we require a minimum tree height of three to achieve a meaningful differentiation between the models, i.e. $|\text{children}_i^{v_m}| > 0$ for at least three nodes. However, we do not encode the hub as a special case of a tree, since the hub encoding is more robust to noise edges. Analogously, we require $E_i^C \leq |E_i^{C_0}|/2$ for the tree structure.

Sparse (○)

A sparse graph is the opposing model to a clique. Every existing edge increases encoding cost. In Figure 4.1 the sparse structure is a planar graph that would be poorly encoded by a clique or a tree structure. In an ideal sparse structure i there are $|E_i^0| = (|V|^2 - |V|)/2$ possible edges. With $|E_i|$ being the number of existing edges above the upper diagonal

$$CC_i = |E_i^0| \cdot H(|E_i|/|E_i^0|). \tag{4.2}$$

Adding the edge probability demands $\frac{1}{2} \cdot \log_2 |V_i|$ bits as further parameter costs. Analogously to the clique model, we add the constraint $|E| < |E^0|/2$.

4.2.2. Graph Compression Schema

Having defined the compression of the graph structures we now define the complete MDL schema for losslessly encoding the full graph. With knowledge about the code book of structures and the encoding schema the full decoding is possible, e.g. after transmitting the data over a channel. The total size mdl_G of the graph after compression is the sum of the cost of the compressed structures $C_i = CC_i + PC_i$ as defined above, together with:

1. **Full graph dimensionality:** the number of nodes of the complete graph $G = (V, E)$ is represented by a natural number of unknown length. It is best compressed using $\log_2^* |V|$ bits (c.f. [Ris07; LC78] for $\log_2^*$). Nodes without any edges belong to no structure since no connectivity information is given. They are included implicitly by sending the graph size.

2. **Number of compressed structures:** Since n represents the upper border for the number of structures, k could be encoded using $\lceil \log_2 n \rceil$ bits. However, since $k \ll n$, using $\log_2^* k$ bits for transmission is obviously cheaper.

3. **Edges connecting structures:** Until now, we only discussed edges in structures. We store information about additional edges *between* structures $E_\times$ by entropy encoding the area of the adjacency matrix lying outside the

subgraph structures. Again, due to symmetry we only consider the upper triangular matrix except the diagonal. The compression requires

$$|E^0_\times| \cdot H(\frac{|E_\times|}{|E^0_\times|}) \text{ bits, with } |E^0_\times| = \sum_{V_i \subseteq V} |V_i| \cdot (|V| - |V_i|). \qquad (4.3)$$

Our graph compression schema mdl_G serves as objective function that has to be minimized by our partitioning and structuring algorithm MeGS.

4.3. Algorithm MEGS

Our proposed algorithm MeGS (Partitioning Meaningful Subgraph Structures using Minimum Description Length) follows a top-down split-and-merge pattern: The overall objective function that is minimized is the size mdl_G of the compressed graph. Starting with the complete graph, MeGS performs a split of a structure into two substructures if reducing MDL. Analogously, structures are merged if reducing MDL. Then, the nodes are assigned to a different structure if reducing MDL. Split, merge and assignment step are iterated until MeGS converges, i.e. neither split nor merge nor assignment reduce MDL any further. We now present the algorithm MeGS[2] (Algorithm 4.1) as well as fast heuristics to build each structure. When referring to time complexity here, we consider the worst time complexity.

Input is a graph $G = (V, E)$. Output are (1) the optimal number k of structures, (2) the optimal type t_i of each structure i, (3) the structures $G_1, \dots, G_k$, (4) the set of edges $E_\times$ between the structures and (5) the length mdl_G of the losslessly compressed graph in bits.

From the compression schema it follows that the objective function is minimized if the structures have fewer edges connecting each other than edges in each structure. We exploit this for initialization such that nodes connected to each other by edges are arranged closer in the permuted adjacency matrix. A *BFS-ordering* permutes the initial adjacency matrix in $O(|E|)$: the child nodes of the first node in A are arranged directly after their parent node in A. If the parent

[2]We provide an implementation of MeGS at `https://s.goebl.net/research`.

Algorithm 4.1: Algorithm MeGS

```
Input: $G(V, E)$
Output: $(k; t_1, \dots, t_k; G_1, \dots, G_k; E_\times; mdl_G)$
mdl_G ⟵ ∞
initialize using BFS-sort
while $mdl_G$ not converged do
    numParts ⟵ 2                                                    ⎫
    repeat                                                           ⎪
        foreach structure $G_i$ do                                   ⎪
            $G'_i$ ⟵ BFS-sorted $G_i$                                ⎪
            split $G'_i$ into numParts equal partitions $p_j$        ⎪
            foreach $p_j$ do                                         ⎬ Split Step
                $G''_i \longleftarrow (p_1, \dots, p_{j-1})$         ⎪ ▷ Section 4.3.1
                $G''_{i+1} \longleftarrow (p_j, \dots, p_{numParts})$⎪
            end foreach                                              ⎪
            lowest $mdl_G$ (if any) → new $G_i$ and $G_{i+1}$        ⎪
        end foreach                                                  ⎪
        numParts++                                                   ⎪
    until last for loop resulted in split or $|p_j| < 4$             ⎭
    repeat                                                  ⎫
        foreach $(G_i, G_j)$ do                             ⎪
            merge into new $G'_i$                           ⎪
            if $mdl_G$ decreases then                       ⎬ Merge Step
                replace $(G_i, G_j)$ by $G'_i$              ⎪ ▷ Section 4.3.2
            end if                                          ⎪
        end foreach                                         ⎪
    until no more merges possible                           ⎭
    foreach node $n \in V$ do                                       ⎫ Assignment Step
        assign $n$ to optimal structure $G_i$ (lowest $mdl_G$)      ⎬ ▷ Section 4.3.3
    end foreach                                                     ⎭
end while
```

node is connected to more than one child node, they preserve their original ordering. Again, the child nodes of the former first child node are arranged accordingly after the last former child node. This produces a stepped pattern in A. If nodes have already been processed (cyclic edges), they remain in place. Until the objective function mdl_G has not converged, split, merge and assignment step are executed iteratively. By definition these steps can only decrease mdl_G but none can increase it. Convergence is assured.

4.3.1. Split Step

For each structure we start (after BFS sorting each first) with two ($numParts := 2$) equal partitions. If the objective function decreases by the newly created structures, splitting is done. In every loop one partition more is created ($numParts$++), and two structures are build in a one-versus-rest pattern. The lowest mdl_G chooses the best split or that no splitting is done. Due to the complexity of this step, let us consider the toy graph (Figure 4.1) with 82 nodes: At the beginning we have one structure G_1. It is split into 2 equal structures $G_1^{''}$ and $G_2^{''}$ containing each 41 nodes. If this new structuring does not decrease mdl_G, three partitions are build: p_1 contains the first 27 nodes, p_2 the second 27 nodes and p_3 the remaining 28 nodes. Now first $G_1^{''}$ is constructed with the first 27 nodes (p_1) and $G_2^{''}$ with the remaining 55 nodes (p_2 and p_3), then a different $G_1^{''}$ with the first 54 nodes (p_1 and p_2) and $G_2^{''}$ with the last 28 nodes (p_3). The lowest mdl_G defines the split. If no split is achieved, the number of partitions increases in the next repetition. If one or more splits have occurred in a single `repeat`-iteration or the partition size is too small, the split algorithm terminates and outputs the (new) set of structures.

4.3.2. Merge Step

Each structure is tested to be merged with each of the remaining $k-1$ structures. If the objective function decreases merging is done. Due to symmetry there is a maximum of $(k \cdot (k-1))/2$ possibilities. We repeat until no more merging is possible.

4.3.3. Assignment Step

After each split and merge step all nodes are processed in the assignment step: if an assignment to one of the $k-1$ other structures reduces mdl_G, the node is assigned to the optimal G_i such that the objective function is minimized. After all the nodes have been processed in random order, all structures are refined in the next iteration of MEGS until convergence. Note that every node is assigned to exactly one G_i.

4.3.4. Graph Structuring Algorithms

When combining two structures G_i and G_j a new structure G'_i is built in the *merge* step so that $V_i \cup V_j \subseteq V$ $(V_i \cap V_j = \emptyset)$. To find the optimal structure all $|\mathfrak{T}| = 5$ structures of the code book are built. Equally, in the *split* step $(numParts - 1) \cdot 2 \cdot 5$ structures are built. In the worst case (no splits found at all) $G_i/4$ partitions are build for each structure *i*. The *assignment* step integrates a new node into an existing structure (Memoizing saves from rebuilding structures). We now present fast heuristics for each structure type: for scalability merging, splitting and assignment should be close to linear time complexity.

Clique and Sparse

No specific node permutation is necessary for building this structure. The time complexity only depends on the calculation of mdl_G being $O(|E| + |V|)$. By memoizing mdl_G, assigning a new node to a clique or sparse structure requires only looking at the direct neighbors $|V_i| \ll |V|$. Thus, the assignment step is nearly $O(|V|)$.

Hub

Building a hub requires finding the highest-degree node and calculating mdl_G, resulting in $O(|E| + |V|)$. By memoizing the degree of the hub node, the assignment operation is analogous to the clique.

Bipartite Graph

We use the popular MAX-CUT approximation algorithm by [SG76] for construction. It constructs two bipartite sets with an approximation ratio of 0.5 in $O(|V| + |E|)$. When assigning a new node to the bipartite set to which it has fewer edges, we only need to count the direct neighbors of the node. Time complexity is nearly $O(|V| + |E|)$ as well.

Tree

Constructing a tree using BFS-ordering (described above) is $O(|E|)$. Calculating mdl_G determines the time complexity of $O(|V| + |E|)$. Insertion of a node is done by local restructuring of the BFS traversal with only a worst case of $O(|V| \cdot |E|)$ in case the whole BFS-sort needs to be repeated. Mean time complexity is close to $O(|V|)$.

4.3.5. Overall Time Complexity of MeGS

The time complexity of each part of MeGS, of the *merge* step (T_{merge}), the *split* step (T_{split}) and the *assignment* step (T_{assign}) has been discussed and is now put together: let *iter* be the number of iterations until convergence and k the number of structures. For simplicity, we do not consider that k can change in every iteration. Instead, we define and use k_{max} as the maximum value for k during the complete algorithm. The outer loop of MeGS is executed *iter* times. Then, the split step is executed for every structure, requiring $k_{max} \cdot T_{split}$ time. The merge step is processed for every pair of structures, which takes $\frac{1}{2} \cdot k_{max} \cdot (k_{max} - 1) \cdot T_{merge}$. In the worst case a merged structure merges in the next inner `repeat`-loop again and again ..., adding the factor k_{max}. Finally, each node is assigned to each structure for calculation of mdl_G, except the one it has been assigned to in the previous iteration, requiring $(k_{max} - 1) \cdot T_{assign}$ time. Adding the BFS-sorting initialization of $O(|E|)$, the total time complexity of MeGS is

$$T_{\text{MeGS}} = O(|E|) + O(iter \cdot (k_{max} \cdot T_{split} + \frac{k_{max}^2 \cdot (k_{max} - 1)}{2} \cdot T_{merge} + (k_{max} - 1) \cdot T_{assign})) \tag{4.4}$$

4.4. Experiments

For each contribution provided by MeGS we present several experiments on both synthetic and real-world datasets. We demonstrate MeGS′ effectiveness in experiments on the *partitioning* task (Section 4.4.1), on *structuring and visualization* (Section 4.4.2) and on *compressing* graphs (Section 4.4.3). MeGS′ *efficiency* is shown in Section 4.4.4. The experiments cover *all structure types* contained in our code book (e.g., bipartite structure in BRAIN, tree in POWER). For repeatability we provide all datasets under the same link as our implementation. We compare to the state-of-the-art algorithms CXPrime [Fen+13], VoG [Kou+14], Cross-Associations (CA) [Cha+04], Metis [KK99] and SlashBurn [KF11] (all introduced earlier). Not every algorithm covers all of MeGS′ contributions. Therefore, we choose suitable comparison partners for each experiment. E.g., SlashBurn does not partition a graph and is excluded from the partitioning experiments. An overview of which contribution is compared to which algorithm is given in Table 4.1 (marked by ✔). Methods not marked do not provide this feature.

4.4.1. Partitioning

With known ground truth the task is to find a node assignment so that the final partitions match the ground truth. We compare to MDL-based CXPrime and Cross-Associations which are both able to automatically detect the number k of partitions (like MeGS does). We also compare to Metis and provide the correct k as input parameter here. Partitioning quality is defined by the state-of-the-art measurement Normalized Mutual Information (NMI) (cf. Section 2.2.2).

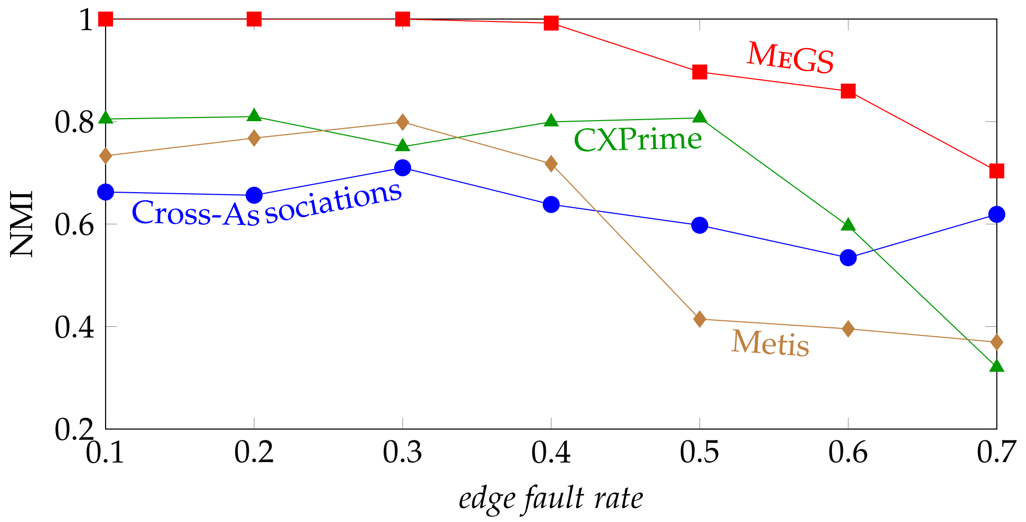
Synthetic Dataset (SYN)

As synthetic dataset (**SYN**) we consider the graph in Figure 4.1. Every structure varies from a *perfect* structure in *faulty* edges: e.g., the green clique shows 52 edges whereas an ideal fully-connected clique with the same number of nodes would display 78 edges. We define this deviation as *edge fault rate*, here it is $\frac{78-52}{78} = 33\%$. Analogously, this applies for all structures as well as the edges *between* structures. This way, we increase noise in each structure and between the structures. The mean edge fault rate of the synthetic graph in Figure 4.1 is 0.4. In this experiment we scale the overall mean edge fault rate from 0.1 to 0.7 by scaling each individual edge fault rate of a structure and between structures by the same factor, thus, making it increasingly more challenging to separate the partitions.[3] The number of nodes remains fixed. CXPrime needs a parameter limiting the interval for searching k. It is set to 10. For Metis the number k of partitions is set to the correct 5. Figure 4.3a (Robustness to noise) presents the mean result of 10 executions for each algorithm. MEGS clearly outperforms all comparison algorithms in partitioning quality and shows robustness to faulty edges. With increasing noise the plot shows a strongly decreasing partitioning quality for CXPrime and Metis. Only Cross-Associations is slightly less influenced by a high noise portion which is probably due to its simple structure model (highly connected vs. unconnected areas in the adjacency matrix).

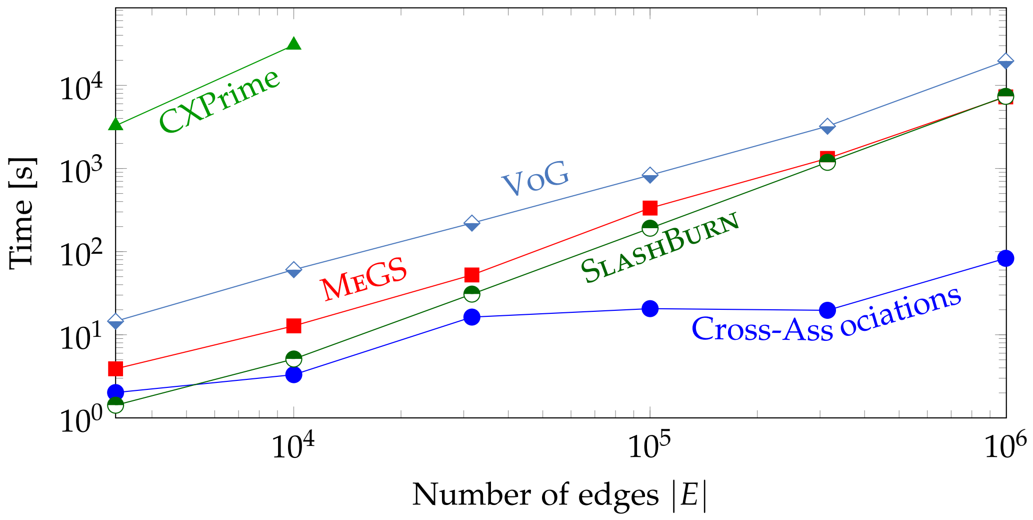
FOOTBALL Dataset

As first real-world dataset we partition the FOOTBALL dataset from [DuB08]. The nodes represent the 115 Division IA (former top level of college football) colleges in the U.S. during regular season Fall 2000. An undirected edge stands for a game between two colleges. The label of a node marks to which of the 12 football conferences a college belongs. We do not use edge weights as also the competitive algorithms are not built for weighted graphs. For CXPrime the

[3]Since an edge fault rate of 0.0 produces separate connected components which the comparison algorithms can not handle, the experiment in Figure 4.3a starts with the mean edge fault rate of 0.1.



(a) Robustness of MeGS and its comparison algorithms



(b) Runtime of MeGS and its comparison algorithms

Figure 4.3.: (a) Robustness when modifying the *edge fault rate* (=rate of edge deviation from ideal structures) on **SYN** dataset (Figure 4.1). *NMI of 1.0 is best.* (b) Asymptotic Runtime when scaling **SYN** in size. Note that MeGS is met by SlashBurn which grows faster than MeGS on the log scale. CXPrime grows too fast to be further plotted. Metis is below 10^0*s*.

upper limit for searching k is set to 15. For Metis k is set to the correct 12. The mean NMI of 10 iterations is given in Table 4.3. With an NMI of over 0.9 MeGS clearly outperforms and partitions the colleges very well into their true football conferences. CXPrime, CA and Metis show increasingly worse results.

Table 4.3.: NMI for FOOTBALL dataset.

MeGS	$\mathbf{0.909 \pm 0.019}$	Cross-Associations	0.721 ± 0.000
CXPrime	0.8233 ± 0.0366	Metis	0.376 ± 0.000

ADJNOUN Dataset

The ADJNOUN dataset (ADJacent NOUNs) from [DuB08] is another real-world dataset. It contains word adjacencies of common adjectives and nouns from the novel *"David Copperfield"* by Charles Dickens. The 112 nodes of the dataset represent the adjectives and nouns of the novel. The 425 edges each connect two words if they are adjacent in the novel. As before, we compare MeGS' partitioning of the ADJNOUN dataset to CXPrime, Cross-Associations and Metis. Since CXPrime is able to identify structure patterns as well (star-like and triangle-like, which corresponds to MeGS' hub and clique) we also compare its findings to MeGS. Since this dataset does not provide a ground truth for evaluating partitioning quality, naturally, the comparison is not as easy as before, and we have to look into the data for assessment. All results are shown in Figures 4.4 and 4.5. For visualization the nodes are arranged in a force-directed layout [Dwy09] where, generally speaking, connected nodes are closer to each other than unconnected nodes. For MeGS the nodes of the graph are labeled with their nouns and adjectives (Figure 4.4). MeGS finds two partitions (blue and red/orange). The orange/red partition is identified by MeGS as a hub: the noun *"man"* (red) serves as hub node and the orange nodes as spokes. This is definitely reasonable and being a common word not surprisingly it is paired with many descriptive adjectives to *"old man"*, *"poor man"*, *"young man"* and others. Figure 4.5a also shows the adjacency matrix permuted by MeGS: the hub node *"man"* is connected to nearly all its spoke nodes. The blue partition is identified by MeGS as sparse structure. The force-directed graph layout stresses that the blue nodes are sparsely connected. They are spread out far around the inner core of the graph. These words are less frequently adjacent in the novel. In fact, the combination *"beautiful woman"* never appears in the text. Thus, we can clearly state that the ADJNOUN dataset proves very well how the structures discovered by MeGS enhance interpretation.

In comparison, CXPrime (search interval limited by 10) displays a very unbalanced partitioning with 29% vs. 71% of the nodes (Figure 4.5b). Both partitions are identified as triangle-like which resembles a clique-like structure. In consequence this means that 70% of the words in the novel are rather adjacent to each other. This is a very general conclusion and provides less insights as MeGS' result. Metis (initialized with $k = 2$) and Cross-Associations only provide partitioning. Metis (Figure 4.5d) shows inhomogeneous, highly entangled partitions in the force-directed layout, and does not clearly separate the data. Cross-Associations (Figure 4.5c) produces three unbalanced partitions, the smallest containing only 5 nodes (or 4%). displays a very unbalanced partitioning with 29% vs. 71% of the nodes (Figure 4.5b). Both partitions are identified as triangle-like which resembles a clique-like structure. In consequence this means that 70% of the words in the novel are rather adjacent to each other. This is a very general conclusion and provides less insights as MeGS' result. Metis (initialized with $k = 2$) and Cross-Associations only provide partitioning. Metis (Figure 4.5d) shows inhomogeneous, highly entangled partitions in the force-directed layout, and does not clearly separate the data. Cross-Associations (Figure 4.5c) produces three unbalanced partitions, the smallest containing only 5 nodes (or 4%).

4.4.2. Structuring and Visualization

The following synthetic and real-world experiments evaluate MeGS's contributions to the *identification of structures* and the *enhancement of visualization* of a graph by permuting its adjacency matrix. We compare to the state-of-the-art algorithms CXPrime [Fen+13] and VoG [Kou+14] for structuring, and to CXPrime, Cross-Associations [Cha+04] and SlashBurn [KF11] for visualization. The other algorithms do not provide these features (cf. Table 4.1). For VoG we use the "greedy and forget" heuristic since it produces good results in [Kou+14].

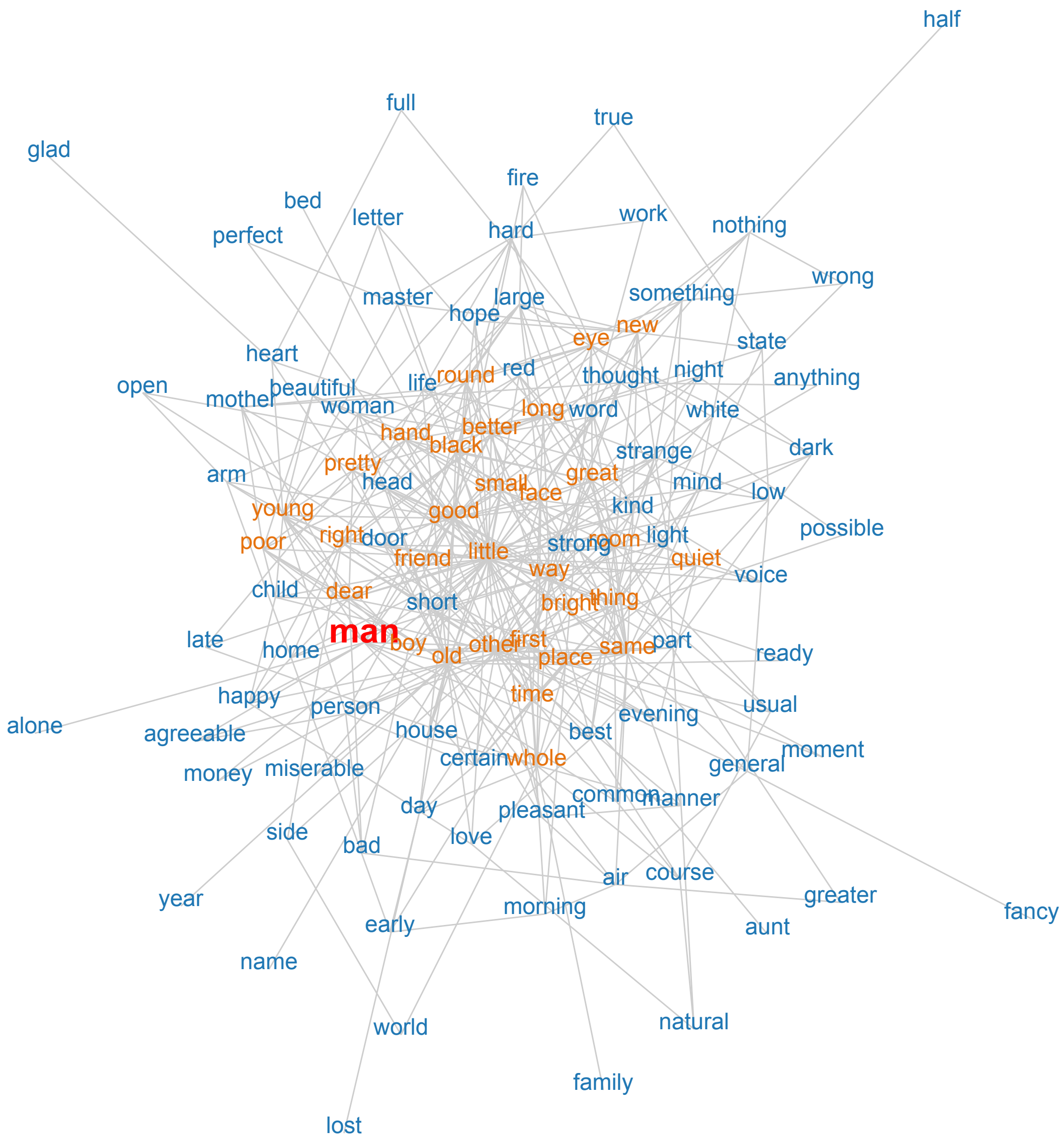
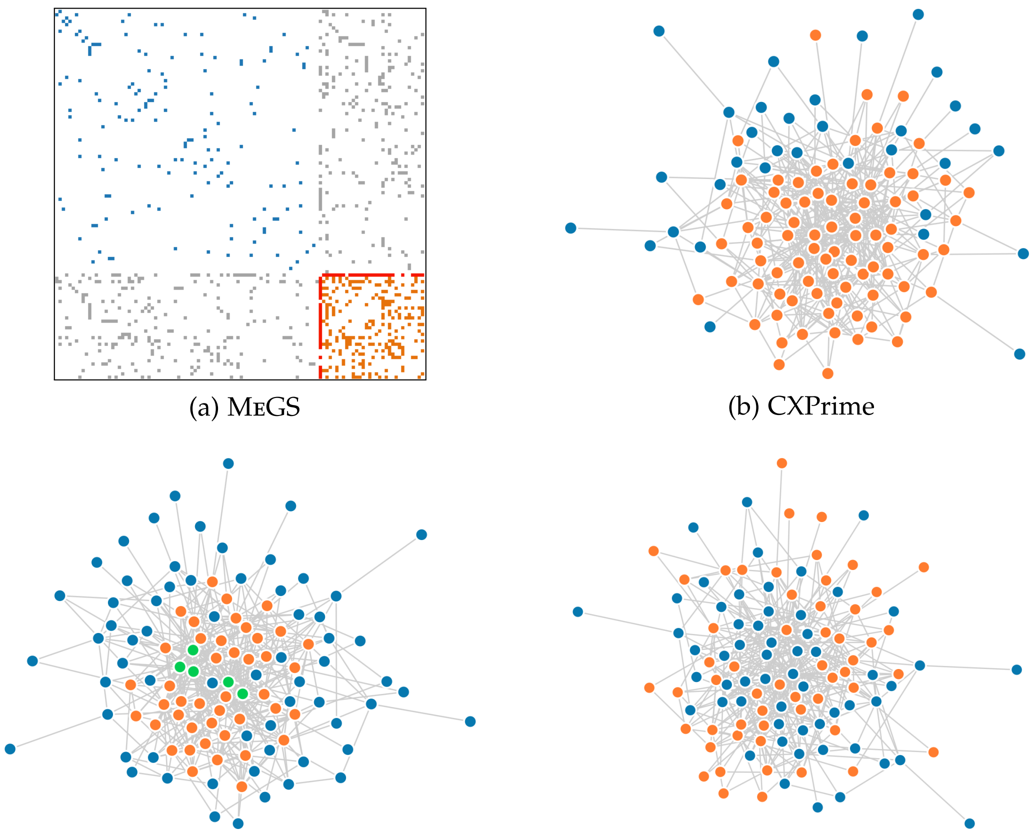


Figure 4.4.: **ADJNOUN dataset:** Force-directed [Dwy09] visualization of MEGS' results. The nodes are labeled with the nouns and adjectives of the dataset. The red node labeled "man" is hub node to the orange spokes. The blue nodes form a sparse structure.



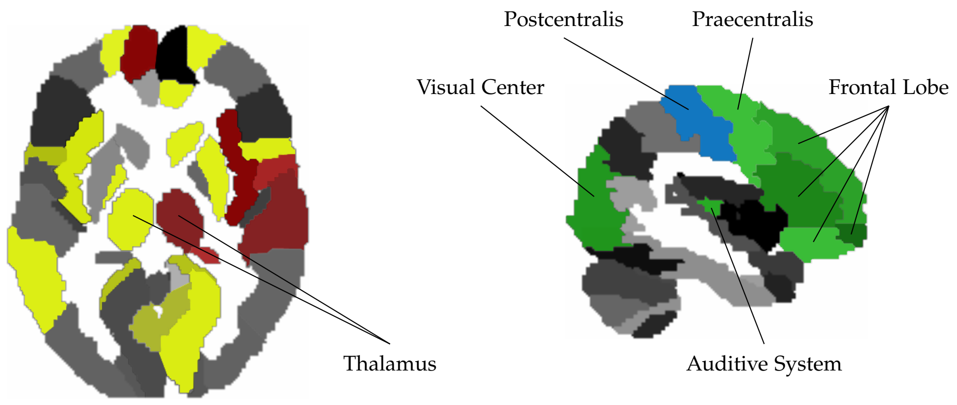
(a) MEGS

(b) CXPrime

(c) Cross-associations

(d) Metis

Figure 4.5.: **ADJNOUN dataset:** (a) Adjacency matrix rearranged by MEGS (cf. Figure 4.4) and force-directed [Dwy09] visualization of the results for (b) CXPrime, (c) Cross-Associations and (d) Metis. Colors in (b)-(d) represent cluster assignments.



(a) Bipartite structure (b) Hub structure

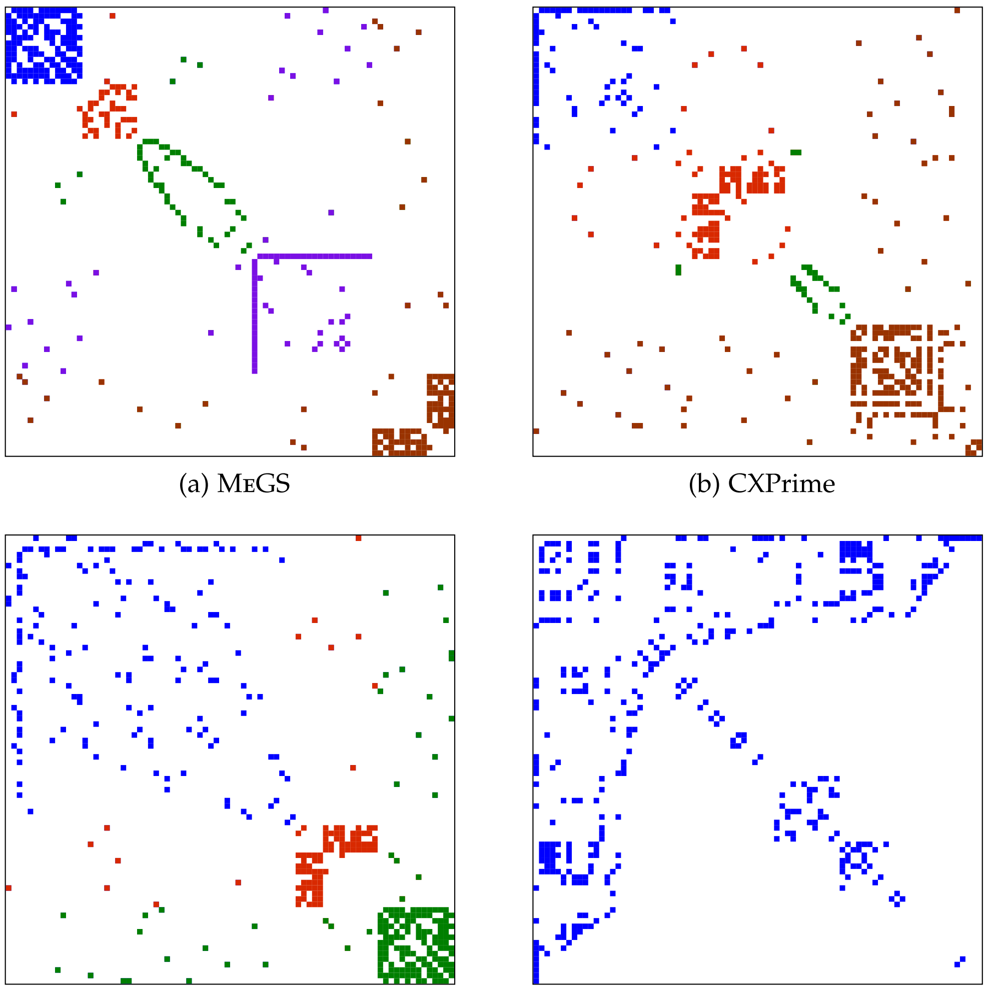
Figure 4.6.: MeGS detects two structures in the BRAIN dataset.

Synthetic Dataset (SYN)

As synthetic dataset we use the example from Figure 4.1 again. For CXPrime the upper limit for search interval is set to 5. Figure 4.7 shows the visualization results. A good visualization and a meaningful structuring is intuitively recognized by the amount of noise and the density of the point clouds in the permuted adjacency matrix. MeGS can identify each structure correctly. Its visualization is most compact with little noise. CXPrime identifies only the hub correctly as star-like structure. The rest is identified as triangle-like structures (i.e. cliques). The visualization is less clear but still gives some interpretable information. Cross-Associations finds three clusters, only two show a compact shape. SlashBurn's ability to visualize power-law degree distributions achieves no compact visualization here. VoG (not featuring visualization) can not extract any structured subgraphs at all: the edges are encoded as error edges altogether.

BRAIN Dataset

The BRAIN dataset [Gün+08] is obtained from a study on somatoform pain disorder (pain without any clinical cause) where patients and control groups got experimentally induced painful and non-painful stimulation on the skin. The dataset is constructed as follows: from a stimulated proband's brain 90



(a) MeGS

(b) CXPrime

(c) Cross-Associations

(d) SlashBurn

Figure 4.7.: **SYN dataset:** Visualization of the adjacency matrix rearranged by the algorithms (a) MeGS, (b) CXPrime, (c) Cross-Associations and (d) SlashBurn. To recognize the different partitions found by the algorithms each partition is colored differently.

time series with 325 time points are recorded by 3d volume functional Magnetic Resonance (fMRI) imaging. According to [Tzo+02], the brain is divided into 90 anatomical regions so that every time series refers to a specific brain region. Here, the values of the time series indicate how active the brain region at a specific time point is. In order to extract a graph structure, an edge is drawn between the brain regions that are showing activity above a certain threshold at the same time point. Thus, the resulting BRAIN graph shows the most active brain regions that share some information with each other.

The processing of the BRAIN dataset by MEGS results in a bipartite structure and a hub. To evaluate this result Figure 4.6 presents the involved brain regions marked with colors. The perspective of the brain image is chosen according to the biggest number of regions that can be displayed in this view. The functions of the brain regions discussed are defined in [Orr08]: Figure 4.6a visualizes the bipartite structure in a top view. The red and yellow areas are its two disjoint subsets. The brain area that is assigned to the bipartite sets (red and yellow) are regions that are connected to memory, emotions and associative procedures. Bipartiteness is an intuitive result. These regions have complexer tasks compared to the brain regions inside the hub and therefore show a higher connected structure: they have to share more information. Figure 4.6b visualizes the hub structure in a perspective on the left side of the brain. The blue area is the hub node (=left postcentral gyrus) and is responsible for somatosensory functions, like the perception of the skin or muscles. Therefore, the selection of the hub node is intuitive: the test persons got experimentally induced painful and non-painful stimulation on the skin which is processed by the somatosensory area by transferring information to the other regions. The green marked areas in the first cluster are mainly brain regions that have sensory functions, like the left and right visual center, the left and right auditive system, and the left and right olfactory system (not marked in the graphic). Additionally, several regions of the left and right frontal lobe (responsible for speech and movement control) and the left and right Praecentralis (also responsible for movement control) are assigned to the hub. The interpretation clearly shows that the structures found my MEGS are meaningful.

Figure 4.8 compares the results. The structures visualized by MEGS are easy to see. CXPrime (initialized to search up to $k = 10$) is correctly able to extract the hub, but not the bipartite structure. Cross-Associations extracts four different structures. However, the visualization shows that this is obviously no reasonable structuring. SlashBurn creates a more compact visualization. The white spaces in the adjacency matrix point out that SlashBurn's technique (extracting highest degree nodes) does not fit to the character of the dataset. VoG is not able to extract structures at all but encodes all edges as noise without model-fitting.

POWER dataset

The POWER dataset (from [DuB08]) contains a graph with 4941 nodes and 6594 undirected unweighted edges. The graph represents the topology of the U.S. Western States Power Grid. The visualization results in Figure 4.9 show: MeGS gives the most compact representation with little noise. The very prominent tree structure (blue) intuitively makes perfect sense for a state-crossing power grid. The sparse structure (red) has too many cyclic edges for a tree. Thus, the red partition is still thinly, yet more densely connected than the rest of the power grid. CXPrime (initialized to search up to $k = 15$) identifies 7 noisy unclear triangle- and star-like that do hardly contribute to interpretation. Cross-Associations provides a very noisy visualization with unclear interpretation. SlashBurn is able to sort out some highly-connected nodes, giving a visualization second compact after MeGS. VoG is not able to identify structures.

4.4.3. Compression

MeGS and its MDL-based comparison methods allow lossless compression of a graph. The experimental datasets have been compressed and the minimal MDL (in bit) out of ten runs of each method is listed in Figure 4.4. MeGS outperforms all comparison methods.

4.4.4. Asymptotic Runtime

To demonstrate the efficiency of MeGS, we scale the synthetic (SYN) dataset up to 1m edges. In Figure 4.3b the first data point represents $10^{3.5}$ edges and scales the example graph by factor 4.5 and so on. The *edge fault rate* (defined above)

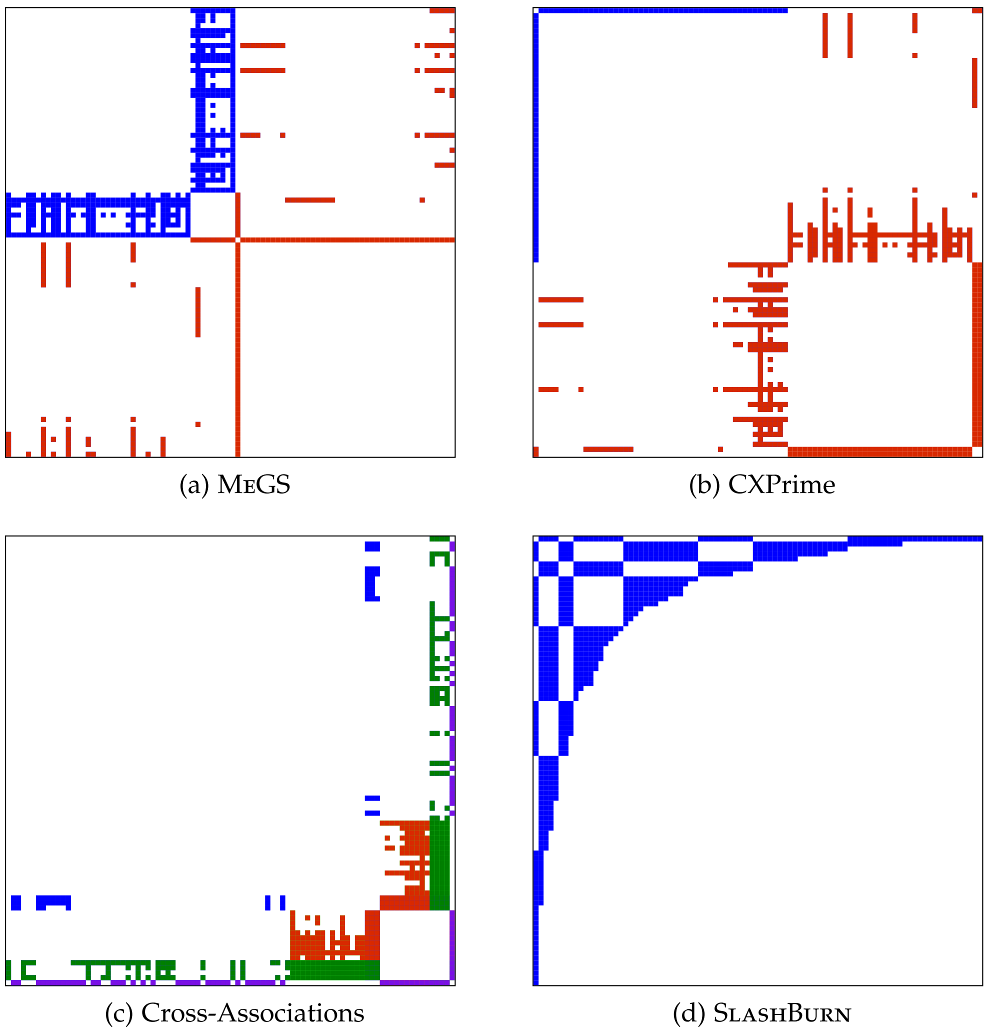
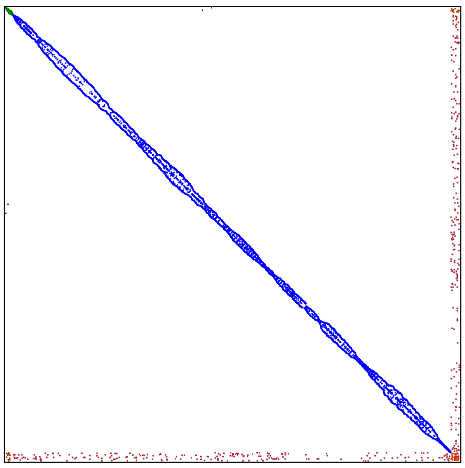
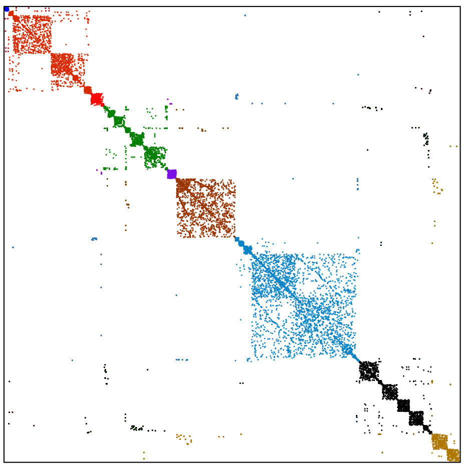
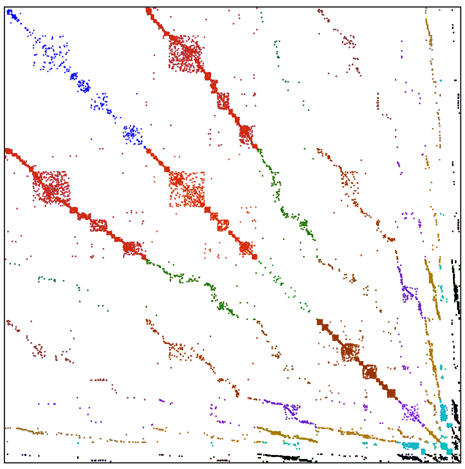

(a) MeGS
(b) CXPrime
(c) Cross-Associations
(d) SlashBurn

Figure 4.8.: **BRAIN dataset:** Visualization of the adjacency matrix rearranged by the algorithms (a) MeGS, (b) CXPrime, (c) Cross-Associations and (d) SlashBurn. To recognize the different partitions found by the algorithms each partition is colored differently.


(a) MeGS


(b) CXPrime


(c) Cross-Associations

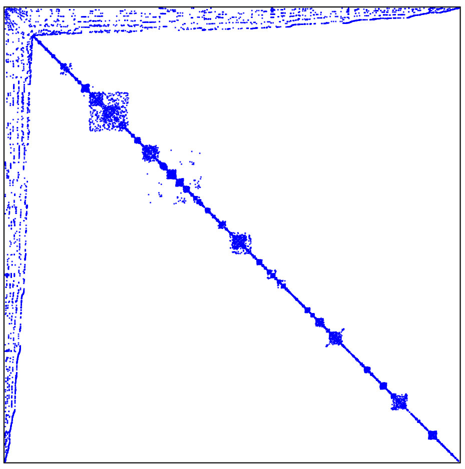

(d) SlashBurn

Figure 4.9.: **POWER dataset:** Visualization of the adjacency matrix rearranged by the algorithms (a) MeGS, (b) CXPrime, (c) Cross-Associations and (d) SlashBurn. To recognize the different partitions found by the algorithms each partition is colored differently.

Table 4.4.: **Graph Compression**: Minimum description length in bits for the datasets ADJNOUN, SYN, POWER and BRAIN. SlashBurn requires block width (b) as parameter for compression: we set $b = 2^i$ with $i \in \mathbb{N}$ chosen to optimize compression. For all datasets the best compression (bold) is achieved by MeGS.

in [bits]	SYN	ADJNOUN	BRAIN	POWER
MeGS	**842**	**2100**	**995**	**76113**
CXPrime	1018	2124	1086	76545
Cross-Associations	2263	4682	1935	208191
VoG	1123	2251	1331	81107
SlashBurn	1959 (16)	3957 (16)	1719 (8)	117857 (64)

in and between structures is fixed to 0.4 as in Figure 4.1. Experiments were performed on a 2.4GHz Intel Core i5 with 8GB RAM. Where applicable we use the parameter settings as in Figure 4.3a, VoG is configured as above. CXPrime and VoG scale worse than MeGS. SlashBurn's curve grows faster than MeGS' and is outperformed for larger graphs where efficiency is indispensable. Cross-Associations is faster than MeGS, however, it can only detect clusters of edges but not structures. Metis is very fast (below 10^0s and thus not plotted: however, Metis is solely a partitioning algorithm and is also missing self-parametrization (k is given as input parameter).

4.5. Conclusion

We define an MDL model based on a codebook of *meaningful graph structures*. Our goal is to identify these structures in a given graph. Discovering structures allows interpretation. We provide our split-and-merge algorithm MeGS to partition a graph into the structures defined in the codebook (which can easily be extended). We also show extensive experiments on both synthetic and real-world datasets. As a result, our proposed parameter-free algorithm MeGS is clearly demonstrated to be effective and efficient in comparison to state-of-the-art algorithms. This applies to all our contributions: graphs are separated into an optimal number of partitions and matched to optimal structures, both minimizing the minimum description length, and thus, the compression length in bits.

5. Spectral Lens: Explainable Diagnostics, Tools and Discoveries in Directed, Weighted Graphs

How can we quickly explain large-scale directed and weighted graphs? We present Spectral Lens (SL) to analyze a variety of real-world networks with both negative and positive edge weights, like DBLP relationships, email communications and Bitcoin trust votings. SL offers value on three levels:

(a) Diagnostics: Spectral Lens combines spectral properties from singular value decomposition to create an SL-Dictionary (SLD) to enhance understanding of any directed, weighted graph.

(b) Tools: the SL-Algorithm (SLA) automatically extracts the top groups of nodes with similar connectivity, finds groups of shared connectivity and detects suspicious behavior in a graph.

(c) Discoveries: Experiments on several real-world networks illustrate the effectiveness of SLA.

Observations from synthetic and real-world networks reveal relations between spectral and graph properties. We show that SLA is highly scalable and linear on the size of a graph. Analyzing a graph with over 2 million edges takes less than 5 minutes.

Overall, SL provides an easy-to-use tool for practitioners to explain a weighted and directed graph quickly, to understand its connectivity and identify regular and anomalous behaviors.

5.1. Introduction

Given a multi-million edge directed, negative- and positive-weighted, uni- or bipartite graph — how do we simultaneously

a) find groups of nodes with similar connectivity patterns (e.g., spokes of a hub, a bipartite set, a clique, etc.),

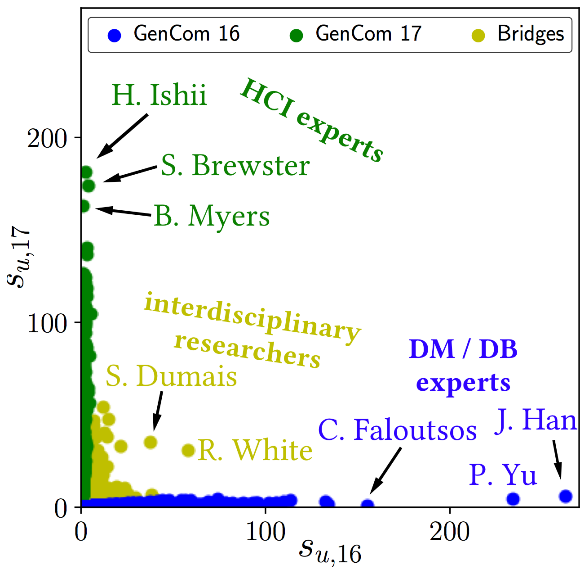
b) identify nodes that share connectivity patterns of two or more groups and connect them like a bridge, and

c) detect nodes with suspicious behavior?

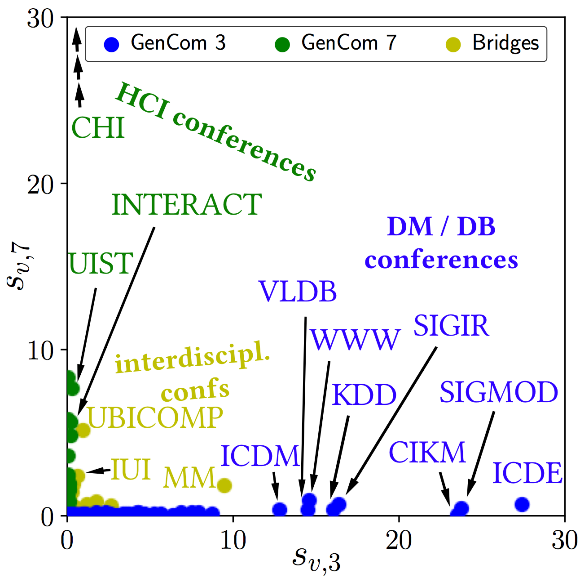
Large scale directed and weighted graphs occur very commonly in the real world. A directed edge can represent an author publishing in a venue, a flight from one airport to another, an indication of trust from one person to another and so on. Weights on these edges can quantify their strength — in a publication graph between authors and conferences, the weight may represent the total number of publications; in the flight network, it can represent the frequency of routes; and in the trust network, it can represent the degree of trust. Moreover, the weight may be negative, e.g., to represent a degree of distrust.

Groups of nodes that have regular connectivity constitute the most commonly occurring patterns in the graph, while those with irregular connectivity occur in case of anomalies and outliers. Identifying both of these groups is useful. For instance, a regular pattern of flights can help to spot densely connected airports, which can be used to optimize air-traffic and scheduling. Notable are nodes belonging to one group but tending to others, such as computer science conferences open for interdisciplinary work: these conferences are *bridges* between different fields. Detecting them can help with one's next decision on where to publish a paper. The irregular patterns in transactions or trust networks can help to spot and prevent fraud.

However, finding these groups of nodes is challenging for several reasons. First, regular or irregular patterns in these networks are not labeled. Second, these patterns can differ from one network to another. Third, there can be multiple regular and irregular patterns that co-exist in the same network.

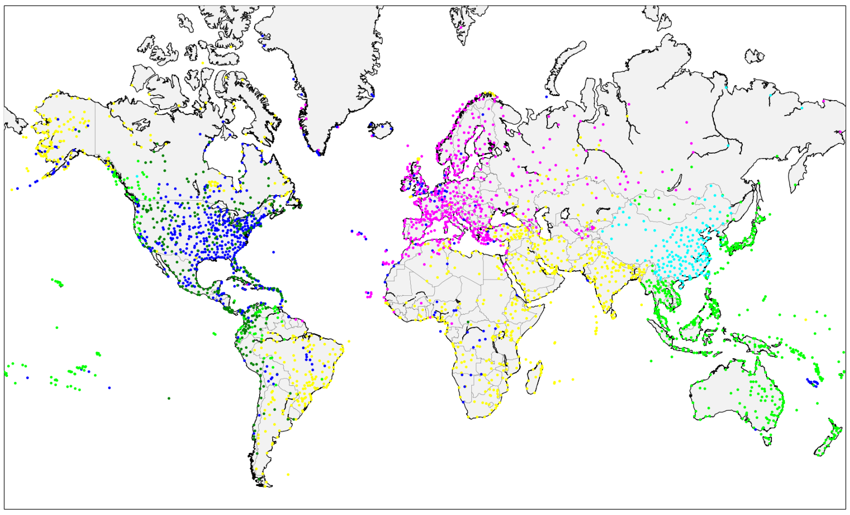


(a) SLA finds experts and interdisciplinary researchers in DBLP

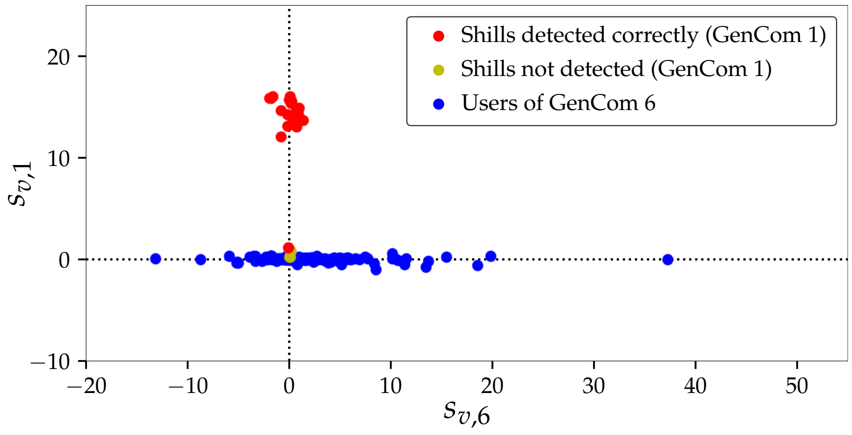


(b) SLA finds fields of research and interdisciplinarity in DBLP

Figure 5.1.: Our proposed algorithm, SLA, explains groups of nodes with similar connectivity (GenComs) and of shared connectivity (bridges): For the DBLP dataset, SLA separates fields and similar researchers and highlights interdisciplinary conferences and researchers.



(a) SLA finds airport patterns in OPENFLIGHTS



(b) SLA reveals shill accounts in BITCOIN-OTC

Figure 5.2.: Our proposed algorithm, SLA, explains groups of nodes with similar connectivity (GenComs) and efficiently identifies anomalous shill accounts from regular accounts: (a) SLA finds airports of similar connectivity (depicted with same color) in the OPENFLIGHTS dataset, (b) SLA isolates accounts that are known to be shills (shown as red dots) in the BITCOIN-OTC dataset.

Moreover, as the graph size increases, the number of possible patterns of the groups increases exponentially. Finally, groups that share the connectivity of different groups are especially hard to assign: they result from the overlap of two or more groups. Even if nodes are part of one group, their extent of closeness to other groups has to be explained for full understanding. None of the existing work identifies and explains both the groups of regular and irregular behavior as well as groups of shared connectivity in directed and weighted graphs.

Therefore, we state the problem definition informally as:

- **Given** a directed and weighted graph,
- **Find** explainable groups of regular connectivity, groups that share connectivity with other groups, and suspicious behavior in the graph spectrum automatically and efficiently.

To address this, we present an unsupervised spectral analysis based technique, called *Spectral Lens (SL)*, that identifies and explains groups of nodes that have regular or irregular connectivity patterns as well as groups that share connectivity patterns with other groups. SL uses *singular value decomposition* as a fast method to decompose large-scale graphs. It finds the connectivity patterns of the left and right singular vectors. SL consists of two parts: **(1)** the SL-Dictionary (SLD) provides a look-up table to understand these patterns and draws conclusions about the graph properties, **(2)** the SL-Algorithm (SLA) automatically spots the top groups of nodes with similar connectivity patterns, groups of shared connectivity and groups with suspicious behavior. We introduce *two novel concepts*:

1. **Generalized Community (GenCom)**: a GenCom is a group of nodes with a similar connectivity pattern. This can be, e.g., a *hyperbolic community* (i.e., a community with a power law degree distribution; cf. [Ara+14]), a clique or a bipartite set: in a fully-connected clique all nodes share the same neighbors and, thus, have similar connectivity; in a bipartite graph both bipartite sets show similar connectivity since they only have edges to the other set.

Table 5.1.: Real-world networks used in this chapter (all are directed, weighted and publicly available).

Nodes	Edges	Name	Description	Comment
1.8m	3.6m	DBLP	author to conference	bipartite
87k	1.1m	ENRON	email communication	unipartite
63k	243k	LKML	mailing list replies	unipartite
6k	36k	BITCOIN-OTC	trust or mistrust	rated
3k	37k	OPENFLIGHTS	flight connections	unipartite

2. **Bridge**: a *bridge* is a group of nodes that share the connectivity pattern of two GenComs but do not form a group of their own; nodes are part of one of the two groups, i.e., bridges result from the overlap of two or more groups and connect them.

To look at SL *in practice*, let us consider a few examples on real-world networks: we gathered all authors and conferences they have published in from DBLP [1] as of February 2017. In the resultant DBLP dataset (cf. Table 5.1) SLA separates computer science fields and researchers with different publishing behavior as well as interdisciplinary conferences and researchers (Figure 5.1). In DBLP, an edge represents an author publishing in a conference, weighted by the number of publications.

In the OPENFLIGHTS[2] dataset, our algorithm finds regions in the world where airports show similar connectivity to other airports (Figure 5.2a). A directed, weighted edge represents the number of routes from one airport to another.

In the BITCOIN-OTC[3] dataset, SLA identifies shill accounts by separating them from the remaining users in the network (Figure 5.2b). The dataset represents users that can vote from -10 to 10 (edge-weights) for other users to show trust or mistrust. Here a group of 44 users is known to be shill accounts of a single user (ground-truth).

[1] `http://dblp.uni-trier.de`

[2] `http://www.openflights.org`

[3] `https://bitcoin-otc.com/trust.php` from [Kum+16]

Table 5.2.: **The contributions of SL compared to methods of related work.** For Graph Detection / Graph Partitioning cf. [Cha+04; DGK07; DMM03; Gre10; KHC05; YL12; YL13], for Anomaly Detection cf. [AMF10; Hoo+16; Jia+14b; KSS14; Sha+16] and for Graph Summarization cf. [KHC05; Kou+14].

	Scalability	*Weighted Graphs*	*Directed Graphs*	*Community/ Group Discovery*	*Bridge Discovery*	*Anomaly Discovery*
Spectral Lens	✔	✔	✔	✔	✔	✔
Community Detection / Graph Partitioning	✔ / –	✔ / –	✔ / –	✔		
Anomaly Detection	✔ / –	✔ / –	✔ / –			✔
Graph Summarization			✔ / –	✔		
EigenSpokes [Pra+10]	✔		✔	✔		

Spectral Lens *differentiates* itself from the related work presented in Chapter 3 by simultaneously explaining both the groups of regular and irregular behavior, as well as groups of shared connectivity. It does so for the more powerful, general case of directed, weighted graphs. Table 5.2 compares the contributions of SL to the contributions of the most related work.

Our proposed method makes the following contributions:

- **Diagnostics**: Spectral Lens combines spectral properties from singular value decomposition to create an SL-Dictionary (SLD) to enhance understanding of any directed, weighted graph. (Section 5.2).
- **Tools**: the algorithm SLA automatically extracts the top groups of nodes with similar connectivity (GenComs), finds groups of shared connectivity (bridges) and detects suspicious behavior in a graph. (Section 5.3).

- **Discoveries**: Experiments on several real-world networks illustrate the effectiveness of SLA (Section 5.4).

For **repeatability**, we additionally provide all datasets and the code of SLA online (cf. Section 2.4).

5.2. Diagnostics: SL-Dictionary

To quickly gain insights into large-scale networks our proposed method Spectral Lens (SL) consists of two parts: SL-Dictionary (SLD) and SL-Algorithm (SLA). For diagnostics of the spectral properties of a graph we present SLD: it guides the practitioner on how to find generalized communities (GenComs) which are nodes with similar (in- and out-edge-)connectivity, such as hyperbolic communities, cliques, bipartite sets or spoke nodes of a hub. At a glance he or she learns the weight-degree distribution of a GenCom and detects bridges which are nodes that share the connectivity of two GenComs and, thus, connect them like a bridge. In this section, we present SLD and the spectral properties which it analyzes, preceded by the mathematical background of SL and a definition of our notations. SLA, the second part of SL, is presented in Section 5.3.

Table 5.3 summarizes symbols and definitions that are important for this chapter. The background on graphs has been summarized in Section 2.1. Note that our method is able to process directed and undirected, weighted and unweighted graphs that also include self-loops or negative edge weights. For easier notation, unweighted graphs can be expressed by weighted graphs with the weight function $w : E \mapsto \{0, 1\}$, and undirected graphs can be represented by directed graphs (for every $(v_i, v_j) \in E$ add (v_j, v_i) to E).

5.2.1. Mathematical Background: Singular Value Decomposition

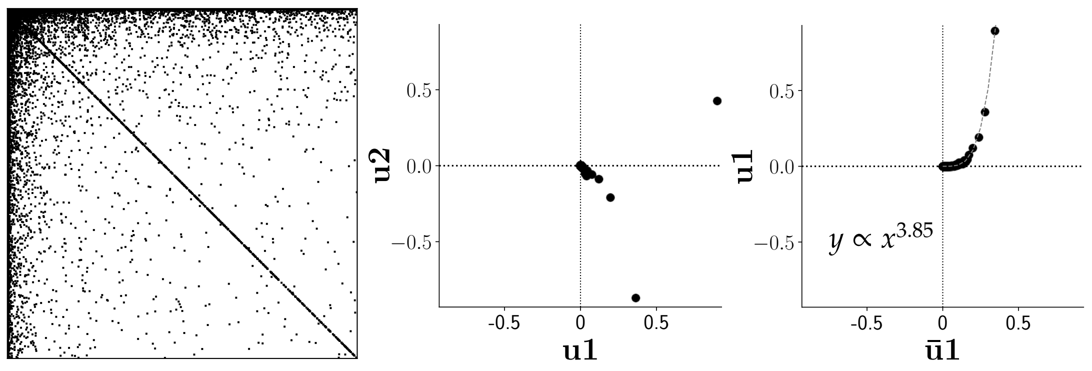
For the look-up patterns in SLD, Spectral Lens analyzes the spectral properties of a graph G. We use *singular value decomposition* (SVD) to compute the spectrum of the adjacency matrix $\mathbf{A}$ of G. Note that $\mathbf{A}$ is not required to be a square matrix

Table 5.3.: Important Symbols and Definitions.

Symbols	**Definitions**
$G = (V, E, w)$	directed, (edge-)weighted graph
$w : E \mapsto \mathbb{R}$	(edge-)weight function
n, m	number of nodes, of edges of G
$\mathbf{u}_i, \mathbf{v}_i$	i-th left, right (weighted) singular vectors
$\bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i$	i-th left, right unweighted singular vectors
$\mathfrak{C}_I, \mathfrak{C}_O$	Set of (in-/out-)-GenComs
$B_{p,q}$	Bridges between GenComs p and q
$\mathfrak{B}_I, \mathfrak{B}_O$	Set of all (in-/out)-bridges

(e.g., for bipartite graphs). General SVD decomposes an $m \times n$ matrix $\mathbf{M}$ into the factorization $\mathbf{M} = \mathbf{U\Sigma V}^T$. Here, $\mathbf{U}$ and $\mathbf{V}$ are $m \times m$ and $n \times n$ matrices, respectively. $\mathbf{U}$'s ($\mathbf{V}$'s) columns (resp. rows) contain the left (resp. right) singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_m$ (resp. $\mathbf{v}_1, \ldots, \mathbf{v}_n$). The diagonal matrix $\mathbf{\Sigma}$ contains the singular values ordered in descending rank. For the spectral analysis of a directed, weighted graph $G = (V, E, w)$, analogously, SVD decomposes the adjacency matrix $\mathbf{A}$ into the $m \times m$ matrix $\mathbf{U}$ and the $n \times n$ matrix $\mathbf{V}$. Since the left and right singular vectors which are related to the top-rank singular values contain the most information about $\mathbf{A}$, SL restricts to the top singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_{k_u}$ and $\mathbf{v}_1, \ldots, \mathbf{v}_{k_v}$ ($k_u, k_v \in \mathbb{N} \wedge k_u, k_v \ll m, n$).[4] The singular vectors constructed in this way represent the weighted graph, thus we also refer to them as *weighted* singular vectors. In addition, Spectral Lens also observes the *unweighted* singular vectors $\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k_u}$ and $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_{k_v}$ obtained through decomposing the original graph G but using edge-degree instead of edge-weight or, formally, through decomposing the graph $\bar{G}$ with the new weight function $\bar{w}(v_i, v_j) = 1$ iff $w(v_i, v_j) \neq 0$. Note that left singular vectors comprise information about connectivity of incoming edges, and right singular vectors information about outgoing edges. This allows to consider both connectivity patterns separately.

[4]Consider, e.g., image compression by SVD to recall this fact.



(a) **Adjacency Matrix** (b) **SL-Plot(u,u)** (c) **BW-Plot(u,u)**

Figure 5.3.: **SL-Plot** and **BW-Plot** of synthetic hyperbolic graph.

5.2.2. GenComs, SL-Plots and BW-Plots

We further consider the already introduced **generalized communities (GenComs)**: it follows from SVD that the *i*-th value of a singular vector represents the connectivity of the *i*-th node in the graph. Thus, nodes with a similar value on the same singular vector show a similar connectivity. Those nodes can be part of the same hyperbolic community, clique, bipartite set, spoke set of a hub and so on. To distinguish this concept from the well-known community concept, we name this node set type *generalized community* or short *GenCom*. In directed graphs nodes can have a different connectivity for incoming and outgoing edges, thus we further distinguish between in-GenComs and out-GenComs.

We define two types of plots using the singular vectors: **Spectral Lens-Plots (SL-Plots)** and **Binary Weighted-Plots (BW-Plots)**. These plots serve as vocabulary which is explained in SLD. Figure 5.3 depicts the plots for a synthetic graph.

1. **SL-Plot(u,u) / SL-Plot(v,v)**: plots of the top left, resp. right, (weighted) singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_{k_u}$, resp. $\mathbf{v}_1, \ldots, \mathbf{v}_{k_u}$, carry information about the connectivity of nodes: if close on the plot, nodes have edges to or from a common subset of nodes and are likely to share the same GenCom.

2. **BW-Plot(u,u) / BW-Plot(v,v)**: plots of top left, resp. right, (*weighted*) singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_{k_u}$, resp. $\mathbf{v}_1, \ldots, \mathbf{v}_{k_v}$, *versus* their *binary*, unweighted

counterpart $\bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{k_u}$ resp. $\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_{k_v}$ (*binary vs. weighted*) explain the weight-degree distribution of a GenCom.

Before we further inspect the spectral properties for SLD, let us look at the amount of spectral information we have to deal with in an example. Figure 5.4 shows only the top-5 left singular vectors ($k_u = 5$) for the ENRON dataset[5] which contains email communication in the ENRON company (messages sent from one user to the other, weights are number of emails). Studying all SL-Plots and BW-Plots requires reading and $4\binom{5}{2} = 40$ plots (for the 19 singular vectors considered for DBLP (Figure 5.1) this would be 684 plots!). Therefore, our algorithm SLA provides an automatic analysis (Section 5.3).

Observation 1 (Number of SL-Plots and BW-Plots). *The top-k_u left and top-k_v right weighted and unweighted singular vectors yield $\binom{k_u}{2} + \binom{k_v}{2}$ SL-Plots and $\binom{k_u}{2} + \binom{k_v}{2}$ BW-Plots.*

5.2.3. SL-Dictionary: Patterns and Rules

We introduce SL-Dictionary (SLD) to understand the spectral patterns that refer to GenComs and their weight-degree distribution. SLD provides three rules for these pattern in a look-up-table (Figure 5.5). For each rule we show a synthetic example graph and a real-world dataset to which the rule applies. All graphs are directed and weighted (Hyperbolic graphs are generated by the RTG graph generator [AF09]).

Rule 1a: Hyperbolic GenCom*: Nodes belonging to the same hyperbolic GenCom follow a power-law pattern in an BW-Plot.*

The synthetic graph is created as a single hyperbolic group so that all nodes belong to the same GenCom. The pattern follows *rule 1a*: in the BW-Plot(u,u) all nodes are lined along a power-law with slope 3.85. The nodes in the LKML[6] dataset belong to the same GenCom and follow a power-law with slope 2.43:

[5] `https://www.cs.cmu.edu/~./enron/`
[6] `http://konect.cc/networks/lkml-reply`

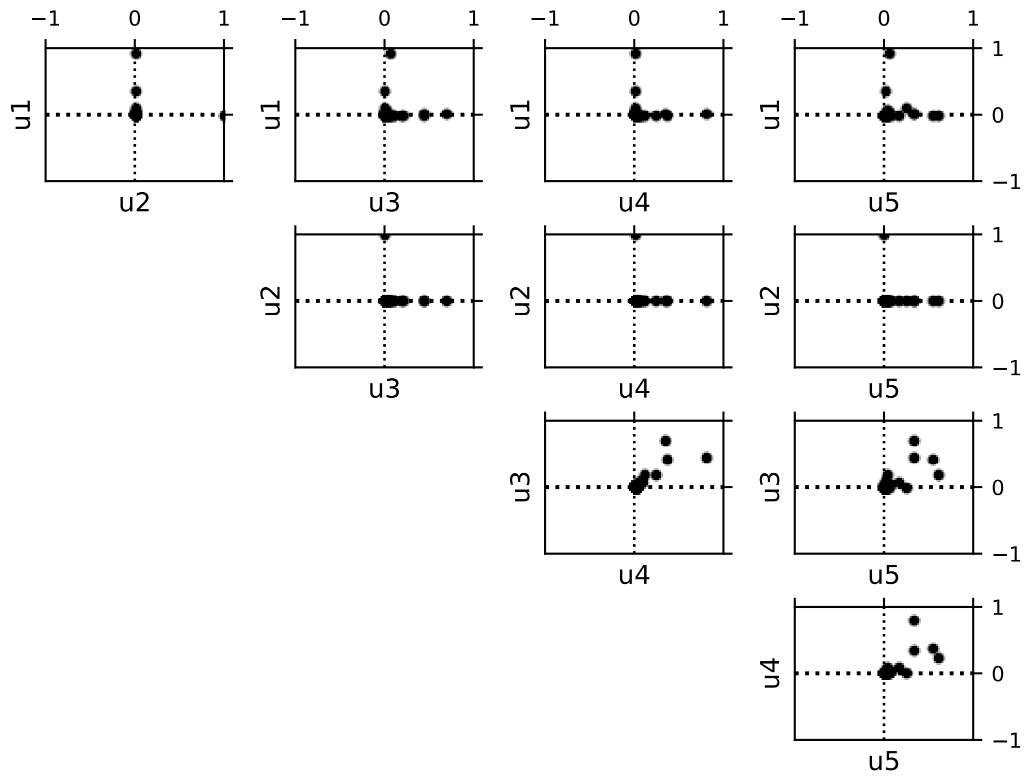
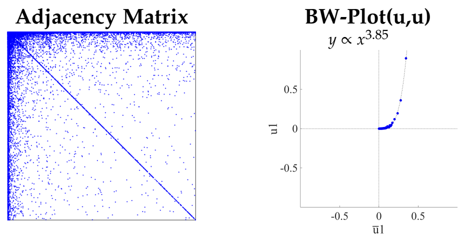
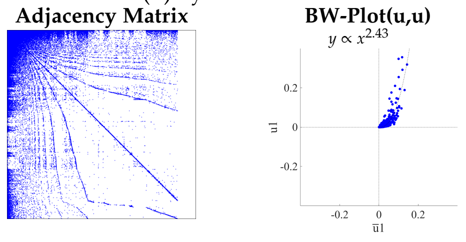


Figure 5.4.: Top-5 SL-Plots(u,u) for the ENRON dataset

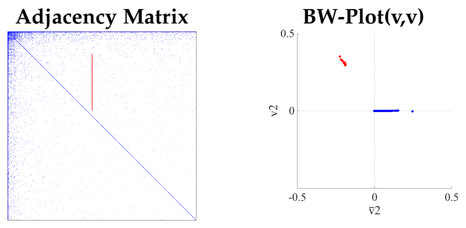


(a) Synthetic dataset

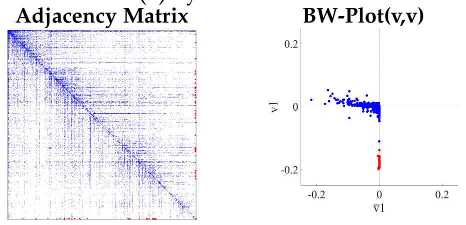


(b) Real-world dataset LKML

Rule 1a: Hyperbolic GenCom

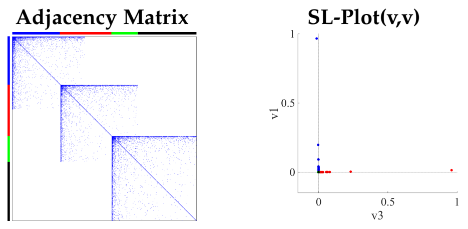


(c) Synthetic dataset

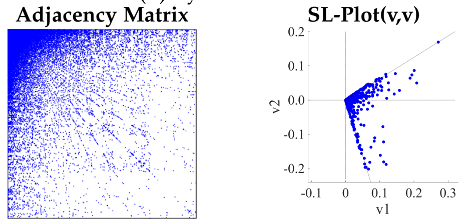


(d) Real-world dataset BITCOIN-OTC

Rule 1b: Non-Hyperbolic GenCom

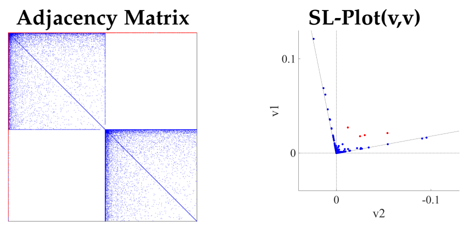


(e) Synthetic dataset

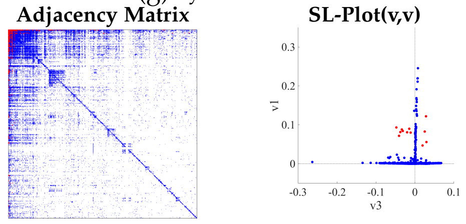


(f) Real-world dataset OPENFLIGHTS

Rule 2: Multiple GenComs



(g) Synthetic dataset



(h) Real-world dataset ENRON

Rule 3: Bridges

Figure 5.5.: SL-Dictionary provides rules for spectral patterns in graphs to quickly explain GenComs and their structure.

many users create a few entries, while few users create many entries. This dataset represents the communication network of the Linux kernel mailing list. A reply from one user to another is expressed by a directed edge, weighted by the number of mails.

***Rule 1b: Non-Hyperbolic GenCom**: Nodes belonging to the same non-hyperbolic GenCom follow a (non-hyperbolic) pattern characteristic to their connectivity in an BW-Plot.*

Now we add a group of 10 nodes with different connectivity to a synthetic hyperbolic GenCom. The group has incoming edges by 2000 followers at an 80% chance (red edges, all edge weights equal 20). In the BW-Plot(v,v) of the second right singular vectors these nodes lie closely together, thus, representing a GenCom with the connectivity pattern of incoming edges of the same weight and from the same group of nodes. Note that the hyperbolic group (blue) is now aligned with the axis. Its power-law curve could be found on the BW-Plot(v,v) of the first singular vector. Similarly, in the BITCOIN-OTC dataset a group of 44 nodes is followed by many other users that give the group high negative trust ratings. These nodes build a very compact GenCom in the BW-Plot(v,v). This small compact GenCom hints to suspicious connectivity.

***Rule 2: Multiple GenComs**: Nodes belonging to the same GenComs align with the same axis in an SL-Plot. Overlap results in tilt from one of the two axes.*

The synthetic graph is constructed by three hyperbolic graphs with some overlap. The sections marked blue and red each contain nodes that share their connectivity (GenComs) and are aligned to the axes in the SL-Plot(v,v). The third GenCom is aligned to another orthogonal axis and, thus, lies here at the origin. Note, that the overlap of an area with lower weights of one subgraph with the higher weight nodes of the other subgraph does not result in a tilt. However, the overlap in the OPENFLIGHTS dataset is strong enough to cause a tilt in the SL-Plot(v,v). Also, other GenComs are not orthogonal to the plotted singular vectors and projected onto them. A problem that is solved by SLA.

***Rule 3: Bridges**: Nodes that connect two GenComs lie between the axes in the SL-Plots representing the two GenComs.*

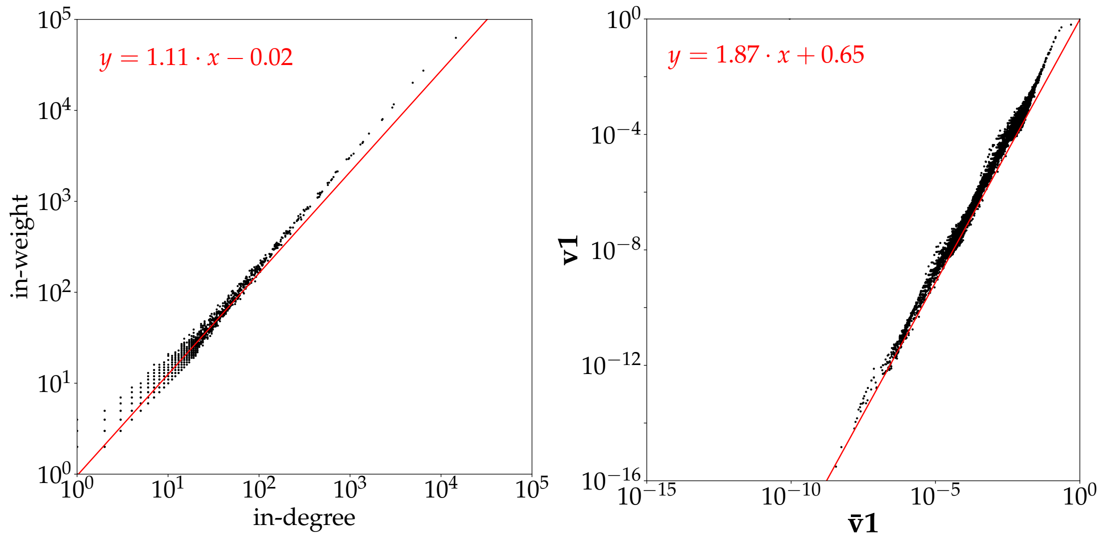
Two synthetic hyperbolic GenComs are now connected by high-degree nodes that show several larger-weighted connections to each other (red edges). In the SL-Plots(v,v), the nodes connecting the GenComs lie between both GenComs. We call these nodes *b*ridges. The ENRON email dataset shows bridges as well: these nodes represent mostly vice presidents of ENRON which receive emails from different parts of the company, which leads to their special connectivity pattern.

The insights gained from SLD allow to make an observation in the singular value decomposition of power law distributed graphs:

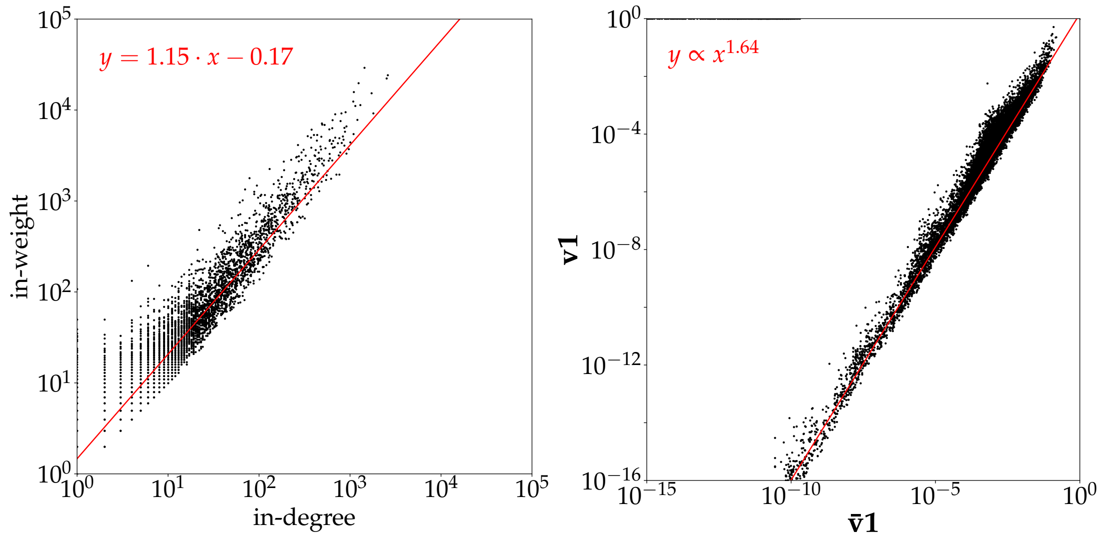
Observation 2 (Correlation of Power-Law Distributions)**.** *If a graph follows a power-law (in/out)-weight vs. (in/out)-degree distribution, then we also observe a power law in the distribution of the nodes in the BW-Plot(u,u) resp. BW-Plot(v,v). The first (right or left) weighted singular vectors form a power law with its unweighted counterpart. (An example is given in Figure 5.6).*

5.3. Tools: SL-Algorithm

With the SL-Dictionary (SLD), the first part of Spectral Lens, we enabled the practitioner to read the spectral properties of a graph from Spectral Lens-Plots (SL-Plots) and Binary Weighted-Plots (BW-Plots). These plots explain generalized communities (GenComs), their weight-degree distribution and bridges. However, we already mentioned the limitations of using SLD alone: the number of plots that need to be taken into account (cf. Observation 1) is far too large for manual analysis, especially for large-scale networks with many GenComs like in the DBLP dataset (Figure 5.1). Another limitation occurs for strongly connected GenComs that tilt in the SL-Plots (cf. Rule 2 in Figure 5.5): since all GenComs are projected onto the 2-dimensional SL-Plot, non-orthogonally tilted GenComs confuse the picture. Finally, for detecting suspicious connectivity patterns in the GenComs the user has to compare all BW-Plots for suspicious weight-degree distribution.



(a) Observation 2 on synthetic graph



(b) Observation 2 on LKML dataset

Figure 5.6.: **If** a graph follows a power-law weight-degree distribution (left column) **then** the first BW-Plot(u,u) resp. BW-Plot(v,v) follows a power-law (right column). A synthetic graph and the LKML dataset together with the corresponding BW-Plot(v,v) illustrate Observation 2.

To overcome these limitations, we propose our algorithm SLA to extract GenComs, bridges and to sort the GenComs according to suspicious connectivity for a given graph (Figure 5.9). To overcome the tilt, SLA uses independent component analysis (ICA).

Formally, we state the problems that SLA solves as :

Problem 5.1: Find *GenComs*

Given

- Directed, weighted graph $G = (V, E, w)$
- Weight function $w : E \mapsto \mathbb{R}$

Return

- Set of in-GenComs $\mathfrak{C}_I$
- Set of out-GenComs $\mathfrak{C}_O$
- Suspicion score $\mathbf{e}_I$ for in-GenComs
- Suspicion score $\mathbf{e}_O$ for out-GenComs

Problem 5.2: Find *bridges*

Given

- Directed, weighted graph $G = (V, E, w)$
- Weight function $w : E \mapsto \mathbb{R}$
- Set of in-GenComs $\mathfrak{C}_I$
- Set of out-GenComs $\mathfrak{C}_O$

Return

- Set of in-bridges $\mathfrak{B}_I$ so that bridges $B_{I_{p,q}} \in \mathfrak{B}_I$ *iff* $(C_p, C_q) \in (\mathfrak{C}_I, \mathfrak{C}_I) \wedge (p \neq q)$.
- Set of out-bridges $\mathfrak{B}_O$ so that bridges $B_{O_{p,q}} \in \mathfrak{B}_I$ *iff* $(C_p, C_q) \in (\mathfrak{C}_O, \mathfrak{C}_O) \wedge (p \neq q)$.

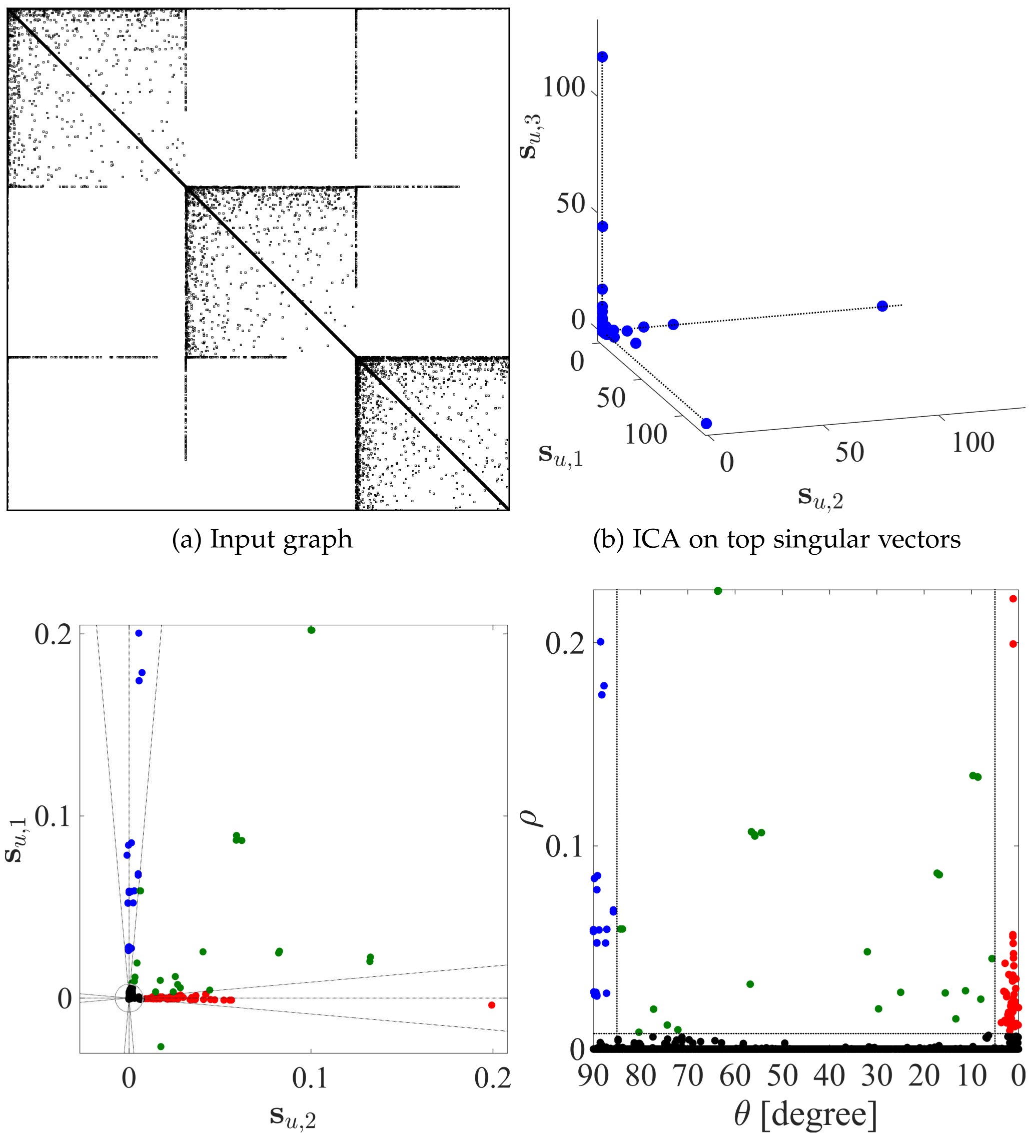
5.3.1. Intuition

To understand the high-level idea of our algorithm, we give an overview of our approach (Figure 5.7): SLA takes nodes and edges of a graph as input. Since the spikes corresponding to the GenComs can be tilted, we apply independent component analysis on the left and right singular vectors (Figure 5.7b). Now each GenCom is aligned to an independent component. We find the optimal number of GenComs as follows: the closer nodes are aligned to independent components the better is their allocation to GenComs. We introduce *thorniness* as measurement for the level of alignment (Figure 5.7d). Note that connectivity patterns can differ for incoming and outgoing edges. It can lead to different GenComs of in- and out-edges, so we do the detection of GenComs for left and right singular vectors separately, i.e., in- and out-GenComs. With optimal (=maximal) thorniness, we assign GenComs and bridges (Figure 5.7c).

In Section 5.3.6 we show that SLA runs in linear time.

5.3.2. SLA Step 1: Find optimal $k_{u,opt}$ and $k_{v,opt}$

Our algorithm SLA takes a directed, weighted graph $G(V, E)$ with weight function $w : E \mapsto \mathbb{R}$ as input. The first part of SLA (lines 1–17) determines the number of top singular vectors we use from SVD. In detail, SLA finds the optimal dimensionality $k_{u,opt}$ and $k_{v,opt}$ for the left and right singular vectors that maximize the mean *thorniness* (see next section). The number of in- and out-GenComs is defined by $k_{u,opt}$ and $k_{v,opt}$. The outer `for`-loop ends its search interval for best thorniness at a maximum dimensionality k_{max}, that we set to 20. For each k the left and right first k singular vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ and $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are calculated. Then independent component analysis (ICA) is applied to the vectors



(a) Input graph

(b) ICA on top singular vectors

(c) Assign GenComs and bridges.

(d) Thorniness

Figure 5.7.: **Illustration of our SLA algorithm**: (a) Input: synthetic graph composed of three hyperbolic subgraphs with added bridges ($n = 17k, m = 12k$), (b) Independent components of top singular vectors, (c) GenCom 1 (blue) and 2 (red) with bridges (dark green) and nodes of third GenCom (black), (d) thorniness: horizontal lines mark θ_0, vertical line marks ρ_0.

Algorithm 5.1: Algorithm SLA

Input: Graph $G(V, E)$ with vertices V and edges E
Output: Sets of in- and out-GenComs $\mathfrak{C}_I$, $\mathfrak{C}_O$, sets of in- and out-bridges $\mathfrak{B}_I$, $\mathfrak{B}_O$ and suspicion $\mathbf{e}_I, \mathbf{e}_O$.

```
// find optimal dimensionalities $k_{u,opt}$ and $k_{v,opt}$:
for $k \longleftarrow 2$ to $k_{max}$                                              ▷ Section 5.3.2
 do
    $tmax_{u,k} \longleftarrow 0$, $tmax_{v,k} \longleftarrow 0$
    $([\mathbf{u}_1, \ldots, \mathbf{u}_k], [\mathbf{v}_1, \ldots, \mathbf{v}_k]) \longleftarrow$ svd$(G, k)$
    $\mathbf{S}_{u,k} \longleftarrow [\mathbf{s}_{u,1}, \ldots, \mathbf{s}_{u,k}] \longleftarrow$ ica$([\mathbf{u}_1, \ldots, \mathbf{u}_k])$
    $\mathbf{S}_{v,k} \longleftarrow [\mathbf{s}_{v,1}, \ldots, \mathbf{s}_{v,k}] \longleftarrow$ ica$([\mathbf{v}_1, \ldots, \mathbf{v}_k])$
    for $p \longleftarrow 1$ to $k$ do
        for $q \longleftarrow p+1$ to $k$ do
            $tmax_{u,k} \longleftarrow tmax_{u,k} +$ thorn$(\mathbf{S}_{u,k}, p, q)$      ▷ Section 5.3.3
            $tmax_{v,k} \longleftarrow tmax_{v,k} +$ thorn$(\mathbf{S}_{v,k}, p, q)$
        end for
    end for
    $tmax_{u,k} \longleftarrow$ mean$(tmax_{u,k})$
    $tmax_{v,k} \longleftarrow$ mean$(tmax_{v,k})$
end for
$k_{u,opt} \longleftarrow \max_k [tmax_{u,1}, \ldots, tmax_{u,k_{max}}]$
$k_{v,opt} \longleftarrow \max_k [tmax_{v,1}, \ldots, tmax_{v,k_{max}}]$
// Assign GenComs and bridges:
$\mathfrak{C}_I, \mathfrak{C}_O, \mathfrak{B}_I, \mathfrak{B}_O \longleftarrow \emptyset$
for $p \longleftarrow 1$ to $k_{u,opt}$ do
    $C \longleftarrow$ induce$(G, \mathbf{S}_{u,k_{u,opt}})$                              ▷ Section 5.3.4
    $\mathfrak{C}_O \longleftarrow \mathfrak{C}_O \cup C$
    for $q \longleftarrow p+1$ to $k_{u,opt}$ do
        $B_{p,q} \longleftarrow$ bridge-ize$(\mathbf{S}_{u,k_{u,opt}}, p, q)$              ▷ Section 5.3.4
        $\mathfrak{B}_O \longleftarrow \mathfrak{B}_O \cup B_{p,q}$
    end for
end for
for $p \longleftarrow 1$ to $k_{v,opt}$ do
    $C \longleftarrow$ induce$(G, \mathbf{S}_{v,k_{v,opt}})$
    $\mathfrak{C}_I \longleftarrow \mathfrak{C}_I \cup C$,
    for $q \longleftarrow p+1$ to $k_{v,opt}$ do
        $B_{p,q} \longleftarrow$ bridge-ize$(\mathbf{S}_{v,k_{v,opt}}, p, q)$
        $\mathfrak{B}_I \longleftarrow \mathfrak{B}_I \cup B_{p,q}$
    end for
end for
// Score suspicion for GenComs:
foreach $C$ in $\mathfrak{C}_O$ do
    $\mathbf{e}_O \longleftarrow$ suspect$(\mathfrak{C}_O)$                                  ▷ Section 5.3.5
end foreach
foreach $C$ in $\mathfrak{C}_I$ do
    $\mathbf{e}_I \longleftarrow$ suspect$(\mathfrak{C}_I)$
end foreach
return $\mathfrak{C}_I$, $\mathfrak{C}_O$, $\mathfrak{B}_I$,$\mathfrak{B}_O$, $\mathbf{e}_I$, $\mathbf{e}_O$
```

in order to rotate them to independent (thus: axis-aligned) components. We obtain the matrices $\mathbf{S}_{u,k}$ and $\mathbf{S}_{v,k}$ with the independent components from the left and right singular vectors. The values of k that maximize the mean thorniness of each pair of left resp. right independent components are chosen as $k_{u,opt}$ and $k_{v,opt}$ (Figure 5.8).

5.3.3. Proposed Measure: Thorniness

In Figure 5.7b we see the independent components $\mathbf{s}_{u,1}, \ldots, \mathbf{s}_{u,k}$ and $\mathbf{s}_{v,1}, \ldots, \mathbf{s}_{v,k}$ (i.e., rotated the singular vectors after applying ICA). Nodes in a GenCom share their connectivity pattern whilst showing a connectivity pattern different than nodes of other GenComs. The lesser two different GenComs are connected, the better the nodes of each GenCom are aligned on a spike. If both GenComs show the same connectivity pattern, we do not see spikes. In this case, both groups resemble one joint GenCom and must not be split. We exploit this fact to determine the optimal number of GenComs. We define *thorniness* as measurement how well nodes of two GenComs are matching the spike-pattern. A *thorniness* of 1.0 corresponds to perfect spikes, a lower value of *thorniness* to less or no spikes at all (= 0). Formally, we define *thorniness* as

$$\texttt{thorn}(\mathbf{S}, p, q) = \frac{|\{v \in V | \Delta\theta_v(\mathbf{S}, p, q) < \theta_0 \wedge \rho_v(\mathbf{S}, p, q) > \rho_0\}|}{|\{v \in V | \rho_v(\mathbf{S}, p, q) > \rho_0\}|} \tag{5.1}$$

where $\theta_v(\mathbf{S}, p, q)$ (in degrees) and $\rho_v(\mathbf{S}, p, q)$ are the θ and ρ values of the polar-transformed orthogonal projection of $\mathbf{S}$ to the vector space with the basis vectors $\{\mathbf{s}_p, \mathbf{s}_q\}$ (Figure 5.7d). We define the distance $\Delta_v(\mathbf{S}, p, q) = \min\{\theta_v(\mathbf{S}, p, q); 90° - \theta_v(\mathbf{S}, p, q)\}$ which is zero if the spikes are aligned orthogonally. If $\Delta_v(\mathbf{S}, p, q)$ exceeds the threshold θ_0 the nodes are not part of a GenCom, but are bridges. Since θ_0 has to be relatively small, we set it to $5°$. For ρ_0 we choose unity distance ($\sqrt{\frac{1}{n}}$). In the same way, bridges are collected (`bridge-ize` in lines 23 and 31).

5.3.4. SLA Step 2: Assign GenComs and Bridges

After step 1, SLA defines the optimal number of in-GenComs and out-GenComs. The second step `induces` a subgraph for each GenCom. Using the memoized

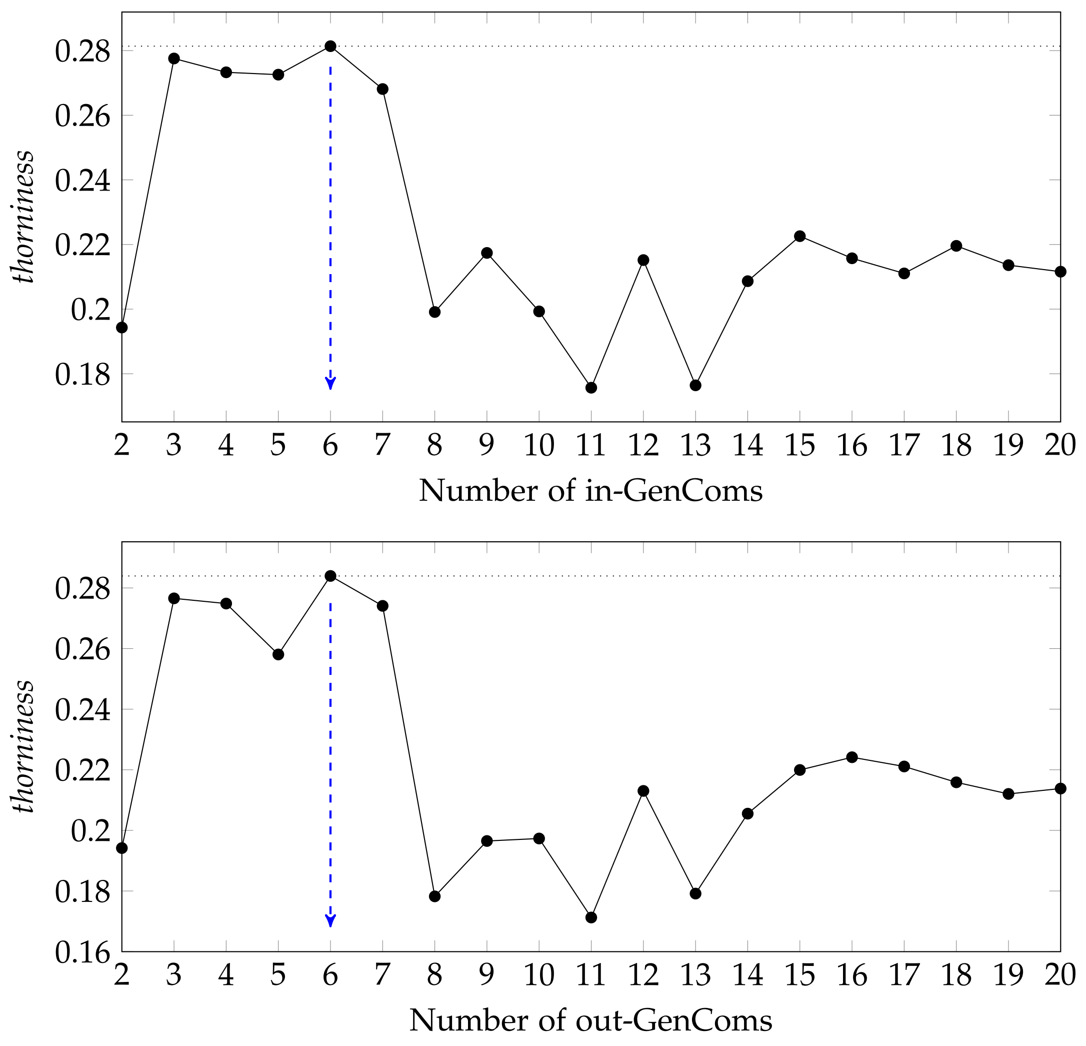
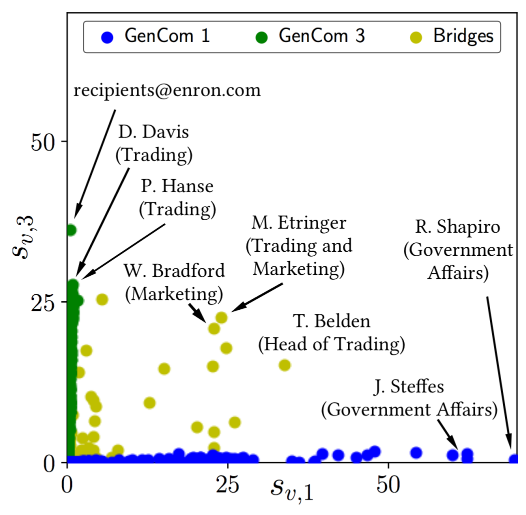
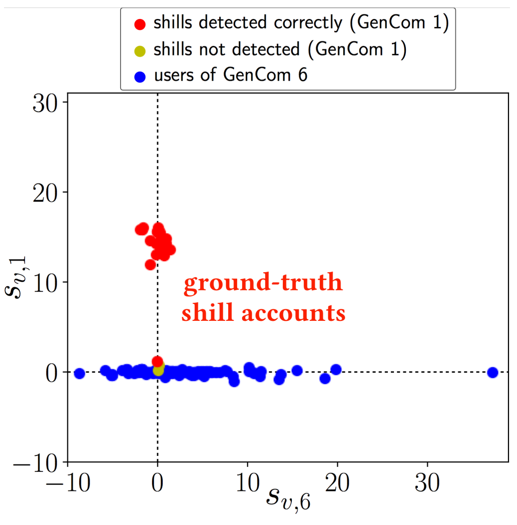


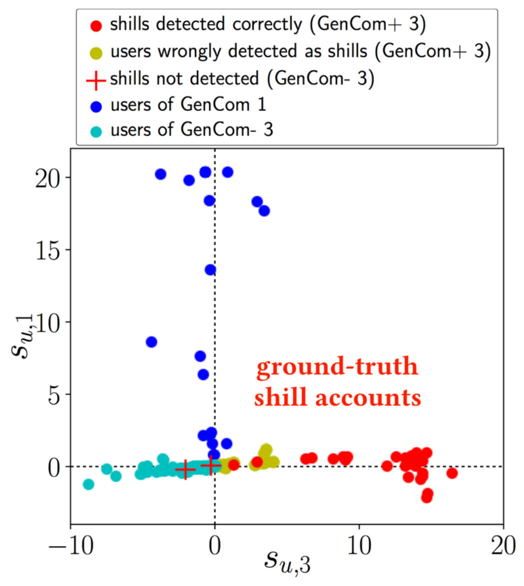
Figure 5.8.: **Our thorniness heuristic gives intuitive results**: the values of thorniness for the OPENFLIGHTS dataset reach their maximum for both 6 in-GenComs and out-GenComs with each GenCom roughly corresponding to a continent.



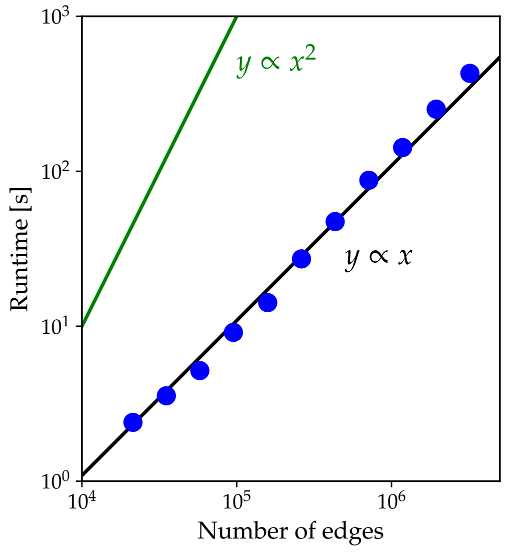
(a) SLA finds GenComs and bridges.



(b) SLA finds suspicious in-GenComs.



(c) SLA finds suspicious out-GenComs.



(d) SLA scales linearly.

Figure 5.9.: Our algorithm SLA detects GenComs, bridges and suspicious behavior: (a) SLA finds GenComs and bridges in the ENRON dataset, (b) SLA identifies the ground-truth shill accounts in the most suspicious in-GenCom and (c) out-GenCom, (d) SLA scales linear: Runtime (in seconds) of SLA (blue) vs. number of edges in graph. Linear (black) and quadratic (green) slopes shown for reference.

results of ICA, every spike ($\leq \theta_0$) becomes a GenCom. SLA assigns nodes to the GenCom as they are defined by *thorniness* (cf. Figure 5.7d). Each in-GenCom is added to $\mathfrak{C}_O$ (line 21) and each out-GenCom to $\mathfrak{C}_I$ (line 29). Finally, SLA collects the bridge-nodes ($> \theta_0$) in $\mathfrak{B}_I$ and $\mathfrak{B}_O$. While for undirected graphs out- and in-GenComs are equal, we have to further split up out- and in-GenComs for graphs with negative and positive weights: negative and positive dimensions of an independent component create separate GenComs: we name them GenCom- and GenCom+. Further, in a positive and negative weighted graph GenCom = GenCom+ $\cup$ GenCom-. For easier understanding, this case differentiation is left out of Algorithm 5.1.

5.3.5. SLA Step 3: Suspicion Score of GenComs

To score the suspiciousness of in-GenComs and out-GenComs, we compare the relation of the number of edges inside the GenCom to the number of all edges in the graph and normalize it by the size of the GenCom. Hyperbolic GenComs with regular behavior show equal relations, whereas typically a suspicious GenCom with fraudsters or shill accounts shows a higher number of edges inside the GenCom (cf. [Hoo+16]). Thus, they have a higher suspicion score. SLA scores suspiciousness (lines 36 and 39) as follows:

$$\texttt{suspect}_\alpha(C) = \frac{\frac{|\{(v_i,v_j)|(v_i,v_j)\in E_{C,\alpha}\}|}{|\{(v_i,v_j)|(v_i,v_j)\in E_\alpha\}|}}{|\{v|v \in C\}|} \tag{5.2}$$

where $\alpha \in \{I, O\}$ is to choose for in- or out-GenComs, $E_{C,\alpha}$ is the set of in or out-edges of the GenCom and E_α is the set of all in- or out-edges in G. After this step SLA terminates.

5.3.6. Scalability of SLA

SLA is efficient for extensive data. In our experiment, we measure the overall running time of SLA for the DBLP dataset. Gradually we remove nodes and their adjacent edges and rerun the reduced dataset. All running times are reported in Figure 5.9d. For reference, we plot the ideal linear growth curve and the ideal quadratic growth curve in log-log-space. The blue dots depict the running time of the executions of the experiment and assemble — especially for larger number of edges — well along the linear growth curve.

5.4. SLA-Discoveries

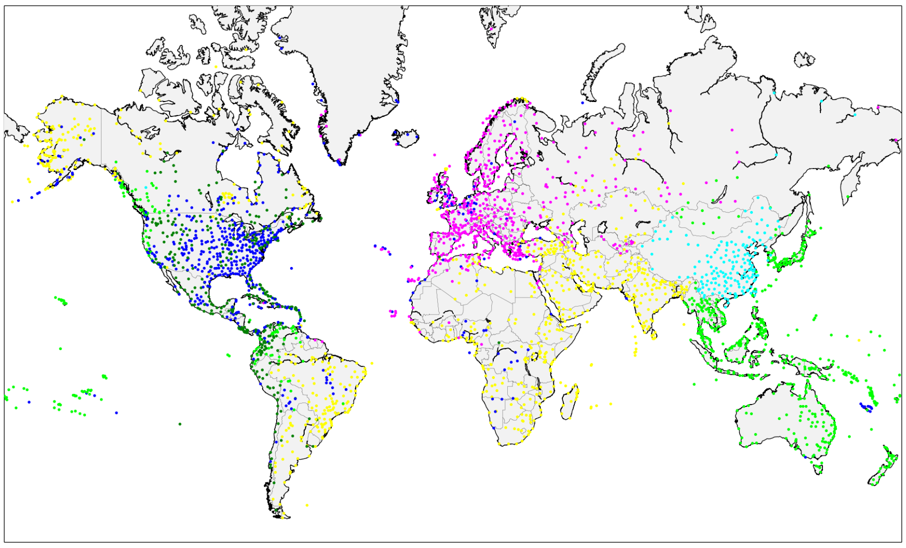
In this section we evaluate the effectiveness of our algorithm SLA on real-world datasets. We reveal discoveries made by SLA in these datasets and show that the algorithm overcomes the limitations to a manual interpretation of the SL-Dictionary. As output SLA gives us GenComs, bridges between them and detects suspicious GenComs. Each GenCom is aligned to an independent component and, thus, the visualization on 2-dimensional plots each containing only a pair of GenComs with their bridges is well explainable and quickly gives insight into large networks. The real-world datasets for demonstrating the effectiveness and efficiency of SLA have been introduced above and are summarized in Table 5.1. We compare the contributions of SLA to the state-of-the-art algorithms in (1) finding overlapping communities BigClam [YL13], AGM [YL12] and COPRA [Gre10] and (2) detecting suspicious behavior FRAUDAR [Hoo+16], LOCKSTEP [Jia+14b], EdgeCentric [Sha+16].

5.4.1. GenComs and Bridges

We present SLA's explainable results by visualizing GenComs with their corresponding independent components and their bridges for the OPENFLIGHTS, DBLP and ENRON datasets. Since only OPENFLIGHTS has a ground-truth in the number of communities we restrict to this dataset for comparison to related methods.

OPENFLIGHTS dataset

This dataset contains flight routes from and to airports all over the world (weights are the number of different routes, i.e. with other layovers). SLA decomposes this dataset into 6 in- and 6 out-GenComs. The airports of the in-GenComs


(a) SLA automatically detects 6 GenComs.

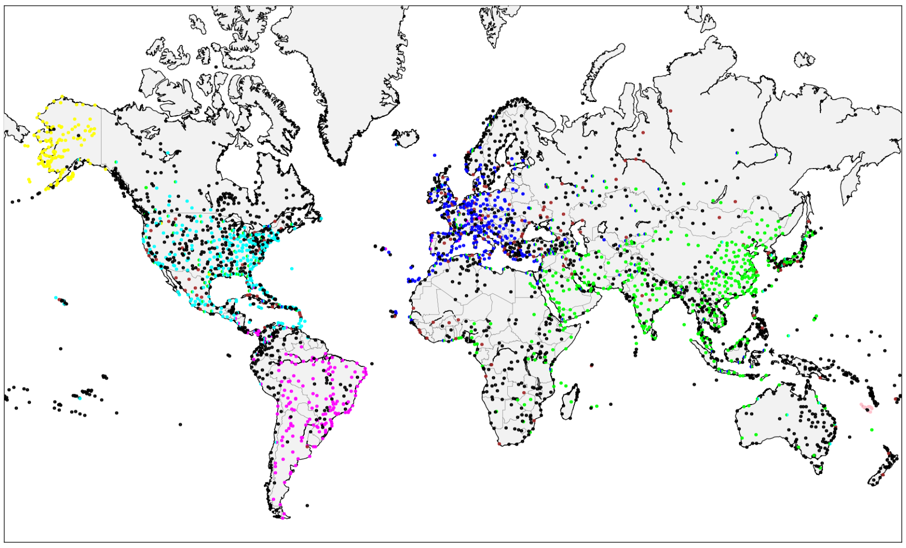

(b) Results by BigClam set to k=6.

Figure 5.10.: **OPENFLIGHTS** dataset: (a) SLA automatically finds 6 groups of airports with similar connectivity (depicted with the same color, bridges not marked for better visualization), (b) BigClam with input parameter $k = 6$ can only assign half of the nodes (non black nodes) and does not further differentiate airports in Asia and Oceania (nodes with overlap of more than 2 communities are depicted in brown color.)

are shown in Figure 5.10a. SLA's choice of 6 GenComs makes sense, they roughly equal the continents: the airports in parts of the world like Europe, Eastern U.S., China or South-Eastern Asia show similar connectivity patterns. The remaining yellow nodes represent regions with much lesser flight routes compared to the other denser regions of the world and, thus, are not separated into different GenComs. BigClam automatically detects 100 communities which does not support visual interpretation of the data. By setting the number of communities as input parameter to 6 (= number of continents) BigClam clusters only 1.7k of the points into communities (Figure 5.10b). BigClam is able to differ airports in Europe and America but can not separate airports in Asia and Oceania. Additionally half of the airports remain unidentified (black dots). AGM automatically detects only one community in the dataset. Even setting the input parameter $k = 6$ does not result in more than one community. The community consists of 95 airports located in Europe and North Africa. Whereas SLA and the other comparison algorithms need a few seconds for this small dataset, AGM requires over 23 hours runtime. COPRA automatically detects 220 communities but does not allow the number of communities to be set as input parameters. Thus, COPRA does not support visual understanding.

This experiment shows that there is *no comparison method* that *automatically* finds an *interpretable* node grouping including bridges/overlaps suitable for interpretation of this dataset.

DBLP dataset

We extracted all conferences with their publishing authors from the DBLP database. We retrieved 1.8m authors and 4.2k conferences. SLA decomposes the dataset into 19 GenComs representing the different fields of computer science and 18 GenComs representing the major research communities. For evaluation, we choose two fields and communities that are of special interest to the reader: Data Mining/Database and HCI (Figure 5.1). The out-GenComs 16 ($\mathbf{s}_{u,16}$) contains the DM/DB-community: the researchers Jiawei Han and Philip S. Yu show high values in their GenCom due to the high amount of publications. In the HCI-community on independent component $\mathbf{s}_{u,17}$ we find the HCI-community with its top-researchers. The bridges (yellow) contains interdisciplinary researchers that work both in DM and HCI: Ryen W. White (founding chair of CHIIR and publications in DM) and Susan Dumais. The evaluations of the in-GenComs well separates conferences in DM/DB and HCI. Note, that DM/DB conferences themselves are not separated which makes sense: the DM community also publishes, e.g. in SIGMOD. With the CHI far outstanding (due to the high number of publications) we find HCI conferences in in-GenCom 7. Interdisciplinary conferences are found as bridges: the closeness of a bridge to a community tells which community it is closer to. E.g., IUI is an HCI conference that is "very interested in contributions from related fields" and explicitly welcomes "solutions from data mining, knowledge representation, novel interaction paradigms, and emerging technologies"[7]. Other in-GenComs represent,e.g., the robotics field (with ICRA on top) or medical cs (with MICCAI on top)

ENRON dataset

The ENRON datasets contains emails from and to employees of ENRON, a company that grew big on energy trading. SLA decomposes the network into 3 in- and 2 out-GenComs: we visualize the in-GenComs 1 and 3 with their bridges (Figure 5.9a): this explains the connectivity pattern of employees who received emails from similar senders. We learn that these GenComs are well separable and comprise employees of different parts of enron. Top recipients in GenCom 1 are employees of Government Affairs, while the other GenCom represents the big Energy Trading part. The bridges that SLA finds make perfectly sense: Tim Belden as Head of Trading, naturally, receives many emails from both GenComs. Also Marketing communicates with different parts of ENRON.

5.4.2. Suspicious GenComs

Scoring the suspiciousness of different GenComs SLA detects fraudulent behaviour in networks, evaluated in the following on the BITCOIN-OTC dataset. This dataset represents users that give a trust or mistrust rating (edge-weight -10 to 10) to each other, a prerequisite for exchanging Bitcoins to money online.

[7]`http://iui.acm.org/2017`

However, this approach attracts dishonest users who create many shill user accounts. These accounts appear as users in BITCOIN-OTC, but are controlled by one user to influence his or her own or other users' ratings. In the additional user comments in BITCOIN-OTC a user who manually controls as many as 48 shill accounts could be extracted as *fraudulent ground truth*.

SLA decomposes the dataset into 9 in- and 5 out-GenComs and automatically recognizes out-GenCom 1 as most suspicious: in fact, this group contains 29 of the shill accounts. We compare the quality of SLA's fraud-detection in terms of F1 Score, precision and recall (Table 5.4) to the recent methods FRAUDAR [Hoo+16], LOCKSTEP [Jia+14b], EdgeCentric [Sha+16]. SLA clearly outperforms all methods in detecting the ground-truth shill accounts in the BITCOIN-OTC dataset since the competitors cannot make use of the negative edge-weight.

Apart from the detection of suspicious behaviour, SLA gives an explanation of its results:[8] for further inspection we visualize in-GenCom1 (equal to +1, since -1 is empty) versus in-GenComs- and + 6 which contains over a third of users and represents normal behaviour (Figure 5.9b). The tight cluster of nodes with very similar connectivity catches one's eye: the users in in-GenCom 1 all show a very similar connectivity pattern, typical for shill users. Also, the second-most suspicious out-GenCom+ 3 shows this tight pattern of nodes of similar connectivity (Figure 5.9c) and explains that the trust votings of this group of accounts do not follow the usual power law (like GenCom 6 in Figure 5.9b or GenCom- 3) which enhances understanding of their suspicious behavior.

The BITCOIN-OTC dataset shows that SLA successfully detects fraudulent behaviour and is able to give an explainable result that allows the practitioner further investigation for insights into the dataset.

5.5. Conclusions

In this chapter, we proposed Spectral Lens which extracts patterns of connectivity and suspicious behavior in the spectral representation of graphs. It consists

[8]Note: since this dataset has positive and negative weights, each GenCom is further distinguished into a GenCom+ and a GenCom-.

Table 5.4.: **SLA performs best** in detecting ground-truth shill accounts in the BITCOIN-OTC dataset compared to recent fraud-detection methods LOCKSTEP, FRAUDAR and EDGECENTRIC.

Method	F1 Score	Precision	Recall
SLA	**0.734**	**0.936**	**0.604**
LOCKSTEP [Jia+14b]	0.046	0.917	0.024
FRAUDAR [Hoo+16]	0.065	0.040	0.167
EDGECENTRIC [Sha+16]	0.046	0.025	0.271

of two parts: the SL-Dictionary (SLD) and SL-Algorithm (SLA). SLD introduces three rules that translate these patterns for practitioners. The scalable SL-Algorithm overcomes limitations to the manual analysis of these patterns; it automatically extracts groups of similar connectivity and scores them for suspiciousness. Spectral Lens works on uni- and bipartite, weighted and unweighted, and directed and undirected graphs.

In summary, our contributions are:

- **Diagnostics**: Spectral Lens combines spectral properties from singular value decomposition to create an SL-Dictionary (SLD) to enhance understanding of any directed, weighted graph.
- **Tools**: our algorithm SLA automatically extracts the top groups of nodes with similar connectivity, finds groups of shared connectivity and detects suspicious behavior in a graph.
- **Discoveries**: Experiments on several real-world networks illustrate the effectiveness of SLA.

SLA is highly scalable and linear on the number of edges in a graph. For future work, it would be interesting to include the analysis of left versus right singular vectors in Spectral Lens: this can give insights into the reciprocity between groups of nodes.

Part III.

Fast and Effective Methods for Parameter-free Clustering

In the previous part, our focus was on the relationships between data entities. Graphs are a valuable tool to model these relationships on an abstract level, and Graph Mining algorithms extract knowledge from these graphs. We have presented our contribution to this field with algorithms that discover patterns in networks, help to understand their structure, identify suspicious behavior — among other features.

We shift our focus from relationships between data entities to finding similarities between the data entities. *Clustering* is the general approach to partition objects into groups or *clusters* based on their similarity. Clustering algorithms assign similar objects to the same cluster and less similar objects to different clusters. This high-level definition applies to a wide area of algorithms and is a central field in Data Mining. Similar to Graph Mining, clustering is a powerful tool to find patterns in massive and complex data. Feature vectors are build to represent complex data like high-dimensional data or data from heterogeneous domains. Among the goals of clustering algorithms is to assign feature vectors to clusters and give insights into the clusters and compare the clusters to each other. Visualizing the data with their clusters is an important step towards *explainable data mining* and avoiding black-box algorithms.

With our first proposed method *FOSSCLU* (**F**inding the **O**ptimal **S**ub**s**pace for **Clu**stering), we find clusters in the lower-dimensional subspace with the highest amount of information. We apply the MDL principle (cf. Section 2.2.1) to guide FOSSCLU to the result without the need of input parameters like the number of clusters or the subspace dimensionality to output. FOSSCLU is able to process complex, high-dimensional data and filters out noisy dimensions with little information. Besides, with FOSSCLU, the user can enforce a low-dimensional output for visual clustering results and create an explainable visualization for a deeper understanding or a starting point for further analytics.

Our second proposed method *INTEGRATE* solves the problem of clustering data with heterogeneous attributes: our technique processes data with numerical and categorical attributes using a unified objective function to assign data points to clusters. Based on the MDL principle, INTEGRATE finds a natural balance between the different domains. Information theory guides the clustering algorithm and requires no input parameters that are usually hard to guess.

The author's previously published work incorporated in Part III This part about contributions to clustering methods with our proposed algorithms FOSSCLU and INTEGRATE is based on and has been published in the following two publications that have been accepted and published in peer-reviewed proceedings of the IEEE International Conference on Data Mining and of the Pacific Asia Conference on Knowledge Discovery and Data Mining [Goe+14; Böh+10]. Additionally, preliminary work for INTEGRATE was conducted in the author's bachelor thesis.

1. S. Goebl, X. He, C. Plant, and C. Böhm. Finding the Optimal Subspace for Clustering. In: *ICDM 14: Proceedings of the 2014 IEEE 14th International Conference on Data Mining.* 2014, pp. 130-139.

2. C. Böhm, S. Goebl, A. Oswald, C. Plant, M. Plavinski, and B. Wackersreuther. Integrative Parameter-Free Clustering of Data with Mixed Type Attributes. In: Advances in Knowledge Discovery and Data Mining. PAKDD 2010, pp. 38-47.

In the following chapters, we present our algorithms FOSSCLU and INTEGRATE.

Paper 1 is a joint cooperation of the author of this thesis with Claudia Plant (University of Vienna, Austria) and Christian Böhm (University of Munich, Germany). The author contributed to the various parts of the papers as well as to the development of the mathematical foundations. He also implemented the algorithm FOSSCLU, conducted the experiments, evaluated the proposed method, and presented the paper. Paper 2 is a cooperation of multiple authors. The author of this thesis designed, and implemented the algorithm INTEGRATE, proofed its effectiveness with multiple experiments and provided content for the presentation.

Structure of Part III In Chapter 6 we present past research related to our proposed methods. Chapter 7 introduces our algorithm FOSSCLU for finding the optimal subspace for clustering. Chapter 8 presents our proposed algorithm INTEGRATE for an integrated clustering of numerical and categorical attributes.

6. Related Work

Our proposed methods FOSSCLU and INTEGRATE for clustering complex data contribute to the following established fields of research: *dimensionality reduction*, *subspace, projected and correlation clustering*, *visualization of subspace clustering results*, and *integrated clustering*. Each area is covered in one of the following sections.

6.1. Dimensionality Reduction Techniques

Dimensionality reduction techniques aim at representing data by a small number of features without much loss of information. Probably the most widely used technique in practice is *principal component analysis* (PCA) [Jol02]. Due to its availability in any statistical package or programming library and its intuitive optimization goal, PCA is often the first choice to plot an unknown data set. PCA projects the data to a low-dimensional subspace, preserving the original variance of the data as much as possible.

Despite the long and ongoing success story of PCA, during the last decades, other feature transformation methods with different optimization goals have been introduced, for example, *independent component analysis* (ICA) [HO00], aiming at identifying statistically independent basis vectors in data; *non-negative matrix factorization* (NMF) [LS00] aiming to decompose data into non-negative, i.e., additive parts; *Isomap* [TSL00] aiming at preserving the geodetic distances. To the best of our knowledge, no feature transformation technique with the objective of preserving the cluster structure in data has so far been proposed.

The approach *LDA-K-means* [DL07] of Ding et al. touches on this challenge. By integrating the supervised dimensionality reduction technique of linear discriminant analysis (LDA) into *k*-means, this technique is a cluster-reinforcing feature transformation technique. However, since LDA is no orthonormal transformation, the resulting space is not a subspace of the original data. Moreover, for K clusters, the dimensionality of the corresponding space is fixed to $K-1$ dimensions. The optimal rigid transform in FOSSCLU is a cluster-enhancing transformation that guarantees orthonormal rotation and projection: what we see is truly in the original data set.

6.2. Subspace, Projected and Correlation Clustering

A variety of specialized techniques exist for detecting high-quality clusters in subspaces. For a comprehensive overview, cf. [KKZ09] and [Sim+13]. Being close competitors to FOSSCLU, we will go into more detail for selected algorithms.

6.2.1. Projected and Subspace Clustering

Projected and *subspace clustering* algorithms search for clusters in axis-parallel subspaces. A benefit of these approaches is their inherent interpretability since the subspaces are represented by subsets of the original attributes.

Aggarwal et al. propose *PROCLUS* [Agg+99] which aims at clustering partitions of the data in cluster-specific axis-parallel subspaces. For initialization, the algorithm greedily selects k medoids as cluster centers from a sample of the data so that the initial medoids are as distant from each other as possible. PROCLUS requires the user to choose the number of clusters with the input parameter k. In a hill-climbing approach, PROCLUS iteratively improves the quality of the medoids. With the average subspace dimensionality per cluster as another input parameter l, the algorithm minimizes the variance of the objects in each cluster in its corresponding subspace. The medoids are iteratively redefined for minimizing the cost function, which could lead to a local cost minimum. Noise is excluded from clusters and returned separately. PROCLUS returns the k clusters with their corresponding axis-parallel subspaces. Disadvantages lie in finding only convex clusters and the difficulty defining the input parameter l.

The grid-based algorithm *CLIQUE* [Agr+98] identifies clusters consisting of dense units in each dimension and builds clusters in axis-parallel subspaces from these regions in an efficient bottom-up style. Clusters are defined as adjacent dense units in each dimension. CLIQUE starts with choosing dense units in a one-dimensional subspace and iteratively adds another dimension to the subspace if the dense units overlap. Due to monotonicity, all clusters identified by CLIQUE in a k-dimensional subspace are also clusters in $(k-1)$-dimensions. Using the MDL principle, subspaces with fewer dense units are pruned away. CLIQUE is deterministic and assumes or requires no underlying assumption of the data distribution, and The user needs to provide two input parameters: ξ defines the grid size, and τ describes the fraction of objects required for a unit to be dense. Both parameters are difficult to guess, and a poor choice of ξ and τ causes CLIQUE to miss clusters that are not caught by in grid.

To overcome the problem of missing clusters, *SUBCLU* [KKK04] proposes density-connected subspace clustering. Using the DBSCAN [Est+96] model, SUBCLU defines clusters as sets of density-connected objects which are maximal w.r.t. density-reachability. The model allows SUBCLU to find arbitrarily shaped and positioned clusters. Like CLIQUE, SUBCLU is able to construct subspace clusters efficiently in a bottom-up style using the monotonicity of density-connected sets: a density-connected set in S is also density-connected in any subspace $T \in S$. The negative case holds as well: if two points are not density-connected in a subspace S, they are not density-connected in any subspace T with $S \in T$. This limits the number of subspaces to consider for the subspace clusters. SUBCLU returns all clusters in all subspaces. In each subspace, SUBCLU finds the same clusters as DBSCAN; clusters of different subspaces can overlap. Also, similarly to DBSCAN, parameters decide on the algorithm's success: ϵ specifies the size of the neighborhood of each point where the density threshold *MinPts* defines the core points for building density-connected sets. However, setting the density threshold gives a bias towards subspace clusters with a certain dimensionality, and can miss high-dimensional clusters.

This bias led to the proposal of the dimensionality unbiased cluster model (*DUSC*) [Ass+07a]. The density threshold influences how well noise is separated and is difficult to set. As a remedy, DUSC introduces an *intuitive density threshold* which defines the density threshold F as a factor relative to the expected density instead of defining an absolute threshold as in SUBCLU. However, this still leaves an input parameter to guess.

6.2.2. Correlation Clustering

Approaches to generalized subspace clustering or *correlation clustering* search for clusters in arbitrarily oriented subspaces of the data space. Since clusters are not limited to reside in axis-parallel subspaces, these methods provide the potential to find more and different clusters. However, for each cluster, these approaches identify an individual subspace in which it is best represented. Our proposed method FOSSCLU determines one joint subspace optimally exhibiting all clusters, where we can study intra- and inter-cluster-relationships between objects in the subspace.

Similar to PROCLUS for finding axis-parallel clusters, *ORCLUS* [AY00] searches for clusters in arbitrarily oriented subspaces. ORCLUS uses an k-means approach but iteratively calculates the distance for the cluster assignment based on each cluster's weak eigenvectors of its local eigensystem. A newly assigned object causes the eigensystem to be recomputed. This approach reduces the variance in each cluster's projection to the weak eigenvectors. Like PROCLUS, the average subspace dimensionality per cluster l and the number of clusters k are input parameters. The algorithm begins with a number of seed clusters higher than k and reaches k by merging clusters pair-wise. Starting with a high number of seed clusters increases the cluster quality but also running time. For finding the best candidates for merging, the eigensystem for the objects of each pair of clusters combined is calculated. In the subspace of the smallest l eigenvectors, the mean squared error from the new centroid decides the optimal pair of clusters to merge. As result, ORCLUS returns the k clusters with their corresponding eigensystems.

Böhm et. al. propose *Computing Correlation Connected Clusters (4C)* [Böh+04b] by extending the axis-parallel projected clustering algorithm PreDeCon [Böh+04a] to arbitrarily oriented subspace clusters. Like PreDeCon, it introduces a parameter δ to modify the distance function and the shape of the ϵ-neighborhood of the underlying DBSCAN-model. In 4C, the parameter δ defines a threshold for weak eigenvectors so that the corresponding distance function receives a higher weight on these eigenvectors. Thus, the ϵ-neighborhood is rotated to match the subspace correlations. For each object, the eigenvalue matrix is modified, and, like PreDeCon, the resulting distance function has to be adapted to be symmetric.

4C returns an arbitrary number of clusters but requires several input parameters. Next to the DBSCAN parameters for the ϵ-neighborhood and its density, the user must provide the threshold δ and the maximal dimensionality λ (without it, 4C would return the full dimensionality as subspace).

As an approach for non-linear clusters, *CURLER* [TXO05] combines the expectation-maximization (EM) clustering model and the ϵ-neighborhood idea to find clusters in arbitrarily oriented subspaces. *Co-sharing* combines microclusters into non-linear clusters.

As a different approach to correlation clustering *CASH* [Ach+] (Clustering in Arbitrary Subspaces based on the Hough transform) applies the idea of transforming objects from the data space into parametrization functions in the parameter space that has been introduced by Hough transform for line detection on pixel images. In clustering, intersection or near-intersection points of hyperplanes (or, more exact: parametrized function) in the parameter space correspond to candidates for correlation clusters. CASH does not consider the distance in the data space and does, thus, not apply the locality assumption. The user needs to specify the parameters m for the number of minimal points for each cluster and s for the number of minimal splits in the iterative refinement of the correlation clusters. CASH is grid-based and uses bisecting search. CASH starts at a $(d-1)$-dimensional parameter space and loops through all dimensions in a fixed order, iteratively splitting each. This approach results in hypercuboids in the parameter space and discards them if they contain less than m objects (the requirement for dense cells of cluster candidates). The hypercuboids containing the most objects are split first. The remaining are returned as $(d-1)$-dimensional subspace clusters, and their objects are removed from the parameter space of other cluster candidates, allowing no overlapping clusters and improving performance. After all cluster candidates have turned into clusters or have been pruned, the next iteration searches for $(d-2)$-dimensional subspace clusters in the $(d-1)$-dimensional data space. Iteratively, the algorithm removes dimensions from the subspace clusters and converges if no further splits are possible. By considering correlations and not using the locality assumption, CASH is robust against noise. However, CASH's worst-case is exponential in the dimensionality d.

6.3. Visualization of Subspace Clustering Results

Only few techniques implicitly or explicitly address the problem that the complex result of subspace clustering techniques is often difficult to interpret. There are two types of techniques:

(1) *Hierarchical methods* inherently providing the benefit of determining and visualizing relationships among subspace clusters, e.g. *HiCO* [Ach+06] and ERiC [Ach+07]. HiCO detects a hierarchy of arbitrarily-oriented clusters of different dimensionality, e.g., two 1D line-like clusters are merged to a 2D cluster if they reside in a common plane. The required agreement in correlation strength and dimensionality of the clusters to be merged are controlled by parameters. The approach ERiC extends HiCO by allowing more complex hierarchical relationships. Both methods require that the subspaces of clusters are hierarchically nested, which is a very special type of relationship.

(2) Two specialized visualization techniques have been proposed for interpreting the result of *axis-parallel subspace clustering* [Ass+07b]. Both techniques allow comparing the clusters w.r.t. space overlap, object overlap, and quality. These techniques intend to assist the user in adjusting the parameter settings of their subspace clustering algorithm DUSC [Ass+07a].

By finding one optimal arbitrarily-oriented subspace during clustering, FOSSCLU naturally facilitates the interpretation of the result. FOSSCLU is not restricted to hierarchical subspace inclusion nor to axis-parallel subspace clusters, and the parametrization is optional.

6.4. Integrated Clustering

Several approaches have been proposed for integrated clustering of datasets with numerical and categorical attributes.

The algorithm *k-modes* [Hua98] transfers the concept of the famous and efficient algorithm *k*-means [Mac67] for numerical clustering data to categorical domains. Instead of using *k*-means' Euclidean distance as dissimilarity measure, *k*-modes defines the distance between two objects as the number of mismatching categorical attributes. While *k*-means assigns objects to the cluster with the closest mean, k-modes assigns them to the cluster with the most similar mode based on the relative frequency for each categorical dimension. Analogously to k-means, k-modes updates the modes and assignments in alternating steps until convergence. Naturally, k-modes shares the restriction for the number of clusters to be determined a priori.

The algorithm *k-prototype* [Hua98] combines k-means for clustering numerical data with k-modes for clustering categorical data in order to cluster mixed-type data. Both dissimilarity measures are combined by summing up the dissimilarity for the individual numerical and categorical dimensions. However, the user has to provide a factor γ to define the weight of the categorical versus the numerical dissimilarity measure.

Yin and Tan [YT05] propose *CFIKP* for clustering large datasets with mixed-type attributes. CFIKP first clusters the dataset using a CF*-tree to store dense regions in lead nodes. Each dense region is represented by a single object in the subsequent clustering step using k-prototype. The problem for selecting the number of clusters remains.

The algorithm *CAVE* [HC07] is an incremental entropy-based method which first selects k clusters, parametrized by the user, and then assigns objects to these clusters based on variance and entropy. Knowledge of the similarity among categorical attributes is required in order to construct the distance hierarchy for the categorical attributes.

The cluster ensemble approach *CEBMDC* [HXD05] overcomes the problem of selecting k but requires a threshold parameter that defines the intra-cluster similarity between objects.

The algorithm *CBC* algorithm [RS06] is an extension of *BIRCH* [ZRL96] for clustering mixed-type data. It uses a weight-balanced tree that needs two parameters, defining the number of entries for (non)-leaf nodes. Furthermore, all entries in a leaf node must satisfy a particular threshold requirement.

Ahmad and Dey [AD07] propose a k-means-based method for mixed-type attributes. However, the process of solving the optimization of the cost function is very complex and, thus, not scalable to large datasets.

[Bro08] uses standard fuzzy c-means on a set of features which is mapped to a set of feature vectors with only real valued components. This mapping is computationally intensive and is designed rather for low dimensional data.

An extension of the cost function of *entropy weighting k-means* [JNH07] to more efficiently specify the inter- and intra-cluster similarities is proposed by the *IWEKM* approac [LC08].

Some papers have focused on avoiding the choice of *k* in partitioning clustering, e.g., *X-Means* [PM00], *RIC* [Böh+06] and *OCI* [BFP08]. However, these clustering methods are designed for numerical vector data only.

7. FOSSCLU: Finding the Optimal Subspace for Clustering

The ability to simplify and categorize things is one of the most important elements of human thought, understanding, and learning. The corresponding explorative data analysis techniques — dimensionality reduction and clustering — have initially been studied by our community as two separate research topics. Later algorithms like CLIQUE, ORCLUS, 4C, etc., performed clustering and dimensionality reduction in a joint, alternating process to find clusters residing in low-dimensional subspaces. Such a low-dimensional representation is extremely useful, because it allows us to visualize the relationships between the various objects of a cluster. However, previous methods of subspace, correlation or projected clustering determine an individual subspace for each cluster. In this chapter, we demonstrate that it is even much more valuable to find clusters in one common low-dimensional subspace because then we can study not only the intra-cluster but also the inter-cluster relationships of objects and the relationships of the whole clusters to each other. We develop the mathematical foundation ORT (Optimal Rigid Transform) to determine an arbitrarily-oriented subspace, suitable for a given cluster structure. Based on ORT, we propose FOSSCLU (Finding the Optimal SubSpace for CLUstering), a new iterative clustering algorithm. Our extensive experiments demonstrate that FOSSCLU outperforms the previous methods even in both aspects: clustering and dimensionality reduction.

7.1. Introduction

Can we obtain both, a meaningful clustering and a projection of the data to 2D or 3D space showing all relevant patterns at first glance? — Despite the vast number of advanced dimensionality reduction and clustering techniques, this is a non-trivial task.

Consider the example dataset in Figure 7.1. The 4D dataset contains three clusters in an arbitrarily-oriented 2D subspace, hidden in two noise dimensions. Figure 7.1(a) displays a 3D scatter plot of the data where one of the uni-modal dimensions has been left out to enable visualization. Figure 7.1(b) shows the result of principal component analysis (PCA) [Jol02] by displaying the scatter plots of the two leading eigenvectors on top and the two minor eigenvectors on the bottom. Clearly, PCA does not preserve much of the information relevant for clustering because the optimization goal of PCA is to preserve the overall variance in data which is here dominated by the noise dimensions. In our example and many real-world datasets, the cluster-separating information is only a small fraction of the overall data variance. Thus there are no separated clusters visible in both projections. Clustering techniques like *k*-means or EM-clustering will never obtain a good result in these subspaces.

During the last decades, the research community has mostly studied the following related problem: Given the clusters, the corresponding subspaces can be easily determined by performing a local PCA on each cluster, and given the subspaces, the clusters can be easily identified. However, if both the clusters and the subspace are unknown, we face a chicken or the egg dilemma. To overcome this, we need to integrate local cluster-wise PCA (or some other dimensionality reduction technique) into the clustering process to detect clusters and subspaces simultaneously as, e.g., in ORCLUS [AY00], 4C [Böh+04b] and CURLER [TXO05]. These methods successfully detect arbitrarily oriented clusters in moderate- to high-dimensional data, but they detect for each cluster one individual subspace in which it is best represented. Allowing an individual subspace for every single cluster might be beneficial for optimizing the clustering quality. However, the result of these algorithms tends to be difficult to interpret since the interesting relationships among clusters remain unclear. In particular, it is not possible to plot the complete cluster structure in one joint low-dimensional space. Figure 7.1(c) displays our example dataset projected to the subspaces determined by the PCA of the blue cluster. One can see that this cluster is axis-parallel due to the decorrelation performed by PCA. The top subfigure corresponding to the leading eigenvectors preserves most of

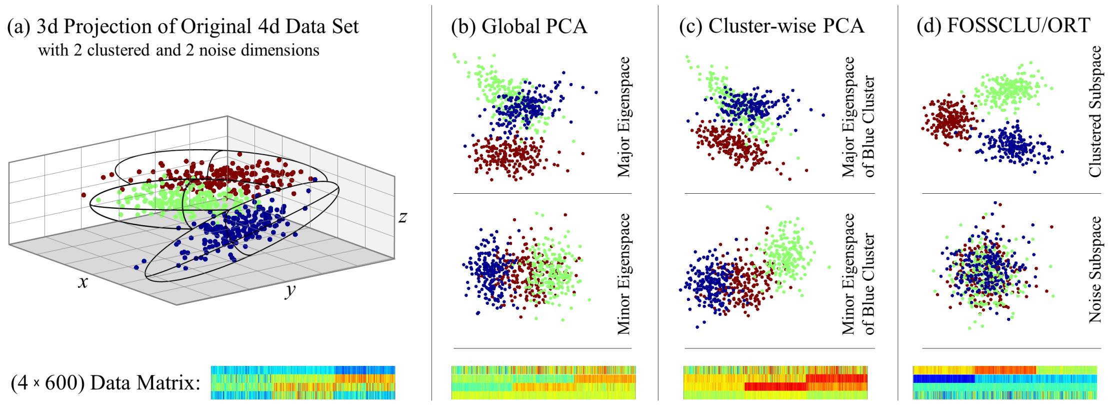


Figure 7.1.: **Determining the Optimal Subspace for Clustering.** (a) Synthetic example dataset. (b) Result of standard PCA: most cluster-separating information is lost. (c) Result of local cluster-wise PCA: Decorrelation of the corresponding cluster, here the blue cluster, but also no preservation of cluster-separating information. (d) Result of FOSSCLU: The optimal rigid transform condenses all interesting information for clustering to the clustered subspace where we have three well separated clusters. The noise subspace is unimodal and can thus be ignored.

the variance of this cluster and is in this sense optimal for its representation. However, this projection does not preserve the information separating the blue cluster from the others. Therefore, it is not suitable for further data mining or interpretation.

Our new algorithm FOSSCLU determines one subspace only. This subspace is optimal for clustering and therefore suitable to plot and analyze the complete cluster structure. The major contribution of our approach is a novel dimensionality reduction technique called the optimal rigid transform (ORT), see Figure 7.1(d). The key idea of ORT is to find an orthogonal pair of subspaces, a *clustered subspace*, ideally containing all relevant information for clustering, and a *noise subspace* for all the remaining variance in the data not relevant for clustering. Among all possible rotations of the data space, ORT determines the rotation that maximizes the cluster quality in the clustered subspace while requiring the noise subspace to be as uni-modal as possible. What exactly do we mean by maximizing the cluster quality? FOSSCLU finds the optimal subspace for *expectation maximization* (EM) clustering. The EM algorithm is among the most widely used clustering techniques and has been voted among the top ten algorithms in data mining [Wu+08]. Figure 7.1(d) (top) displays the clustered subspace found by ORT exhibiting three distinct clusters. See below the noise subspace which is orthogonal to the clustered subspace. It is essential to require the noise subspace to be uni-modal. This ensures that it contains no interesting information for clustering and thus can safely be ignored. See at the bottom line of Figure 7.1 the color-coded data matrix of the original data (a). In original space, some cluster-separating information is contained in every dimension, e.g., the first row of the data matrix representing the x-coordinate roughly separates one cluster from the remaining two. However, there is no single dimension effectively separating all clusters which is found by orthonormal rotation by global PCA (b) and cluster-wise PCA (c). ORT (d) is the only method successfully condensing all the interesting cluster-separating information into a joint low-dimensional space. Note that all clusters are very well separated, even considering only the first dimension of the clustered subspace found by ORT.

While PCA is probably the most widely used feature transformation technique, many others have been proposed, e.g., independent component analysis [HO00], Non-negative matrix factorization [LS00], ISOMAP [TSL00] and many other variants of manifold learning. All these techniques have different optimization goals. However, to the best of our knowledge, none of them aims at finding the optimal subspace for clustering.

We integrate ORT into an EM-style alternating least squares algorithm. Thereby, our algorithm FOSSCLU discovers the clustered subspace and the clusters simultaneously by iteratively optimizing the cluster assignment and the subspace until convergence.

We summarize the following **contributions**:

1. **The optimal rigid transform (ORT)** is a novel cluster-enhancing dimensionality reduction technique. ORT detects the optimal subspace for clustering by maximizing the cluster quality while requiring the orthogonal noise subspace to be as uni-modal as possible. In contrast to existing techniques,

ORT determines one joint subspace for all clusters which greatly facilitates the interpretation of the clustering result.

2. **Accurate Clustering.** Our algorithm FOSSCLU integrates ORT into an efficient EM-style clustering routine. FOSSCLU shows comparable performance to state-of-the-art generalized subspace clustering approaches and even outperforms them. Note that these comparison methods are focusing on clustering only and none of them provide a joint subspace for visualization and interpretation.

3. **Optional Parametrization.** Supported by the minimum description length (MDL) principle, FOSSCLU can automatically discover the number of clusters and the dimensionality of the clustered subspace. However, since users often have good reasons to choose certain parameters, e.g. for application-specific reasons, or to request a specific subspace dimensionality, FOSSCLU can be parameterized.

The remainder of this chapter is organized as follows: The following section elaborates the details of our method FOSSCLU. Section 7.4 provides an extensive experimental evaluation of the visualization and the clustering quality. Section 7.5 concludes this chapter. The related work has already been discussed in Chapter 6.

7.2. FOSSCLU

In contrast to the vast majority of previous methods for dimensionality reduction, clustering, as well as subspace, projected, and correlation clustering, our algorithm FOSSCLU (*Finding the Optimal Subspace for Clustering*) aims at finding a single, unique, m-dimensional, arbitrarily oriented subspace which is optimal for the complete partitioning of the dataset into k clusters by EM-clustering. In our algorithm FOSSCLU, clustering and dimensionality reduction are performed simultaneously in two alternating steps:

(1) each object becomes associated with the best of the k clusters (while fixing a specified dimensionality reduction), and

Table 7.1.: Important Symbols and Definitions.

Symbol	**Definition**
$n \in \mathbb{N}$	Number of objects
$d, m \in \mathbb{N}$	Dimensionality of original (of clustered) space
$k \in \mathbb{N}$	Number of clusters
$X \in \mathbb{R}^{d \times n}$	Matrix containing all data objects in columns
$C_i \in \mathbb{R}^{d \times n_i}$	Matrix of all objects in Cluster $C_i (1 \leq i \leq k)$
$\mu_i \in \mathbb{R}^m$	Center of cluster C_i in clustered space
$\Lambda_i \in \mathbb{R}^{m \times m}$	Covariance matrix of C_i in clustered space
μ, Λ	Center/covar. matrix of *all* data in noise space
$\Sigma, \Sigma_i \in \mathbb{R}^{d \times d}$	Covariance matrix of X (of C_i) in original space
$V \in \mathbb{R}^{d \times d}$	Matrix defining the optimal rigid transform
P_c, P_n	Projection to the first m (last $d - m$) attributes
$G_{p,q}(\theta)$	Givens rotation in plane (p, q) by angle θ

(2) the data space is transformed into the m-dimensional subspace that is optimal for the given clustering.

Our method, therefore, corresponds to the *alternating least squares* (ALS) paradigm. Table 7.1 summarizes the most important symbols and their definitions used in this chapter.

7.2.1. The FOSSCLU Optimization Goal

Our optimization goal is dedicated to the idea of separating the data space $\mathbb{R}^d$ into an orthogonal pair of arbitrarily oriented subspaces: the *clustered* subspace $\mathbb{R}^m$ and the *noise* subspace $\mathbb{R}^{d-m}$. The clustered subspace is selected such that it exactly contains the partitioning of the data objects into k clusters according to an EM-clustering model, while the noise subspace contains no cluster structure. An orthonormal matrix V called *optimal rigid transform (ORT)* defines our pair of subspaces. ORT rotates an object $x \in X$ such that the first m attributes of the resulting object $V^\top \cdot x$ form the clustered subspace. After the rotation, the projection into the first m dimensions is done by the following projection matrix:

$$P_c = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \begin{array}{l} \left.\vphantom{\begin{matrix}1\\0\\ \vdots\\0\end{matrix}}\right\} m \\ \left.\vphantom{\begin{matrix}0\\ \vdots\\0\end{matrix}}\right\} d-m \end{array} \qquad \in \mathbb{R}^{d \times m} \tag{7.1}$$

Analogously the matrix for the projection onto the last $(d - m)$ attributes is defined as:

$$P_n = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \begin{array}{l} \left.\vphantom{\begin{matrix}0\\ \vdots\\0\end{matrix}}\right\} m \\ \left.\vphantom{\begin{matrix}1\\0\\ \vdots\\1\end{matrix}}\right\} d-m \end{array} \qquad \in \mathbb{R}^{d \times (d-m)} \tag{7.2}$$

An object x can be projected to the clustered subspace by $P_c^\mathsf{T} \cdot V^\mathsf{T} \cdot x$.

Our optimization goal is to find the ORT V and a grouping of the objects into clusters $C_1, ..., C_k$ such that

- the objects $P_c^\mathsf{T} \cdot V^\mathsf{T} \cdot x$ in the *clustered* subspace follow a multi-modal distribution with k peaks corresponding to the clusters $C_1, \cdots, C_k$ and
- the objects $P_n^\mathsf{T} \cdot V^\mathsf{T} \cdot x$ in the *noise* subspace follow a uni-modal distribution.

The optimization goal of FOSSCLU is given in the following:

Definition 7.1: Optimization Goal of Fossclu

Given $k, m \in \mathbb{N}$, find an orthonormal matrix $V \in \mathbb{R}^{d \times d}$ and a grouping of the objects in X into k clusters $C_1, \cdots, C_k$ such that the following objective function f (called *Fossclu function*) is minimized:

$$f = -\sum_{1 \le i \le k} \sum_{x \in C_i} \log_2 \left(\frac{n_i/n}{\sqrt{(2\pi)^m |\Lambda_i|}} \mathrm{e}^{-\frac{1}{2}(x^\mathsf{T} V P_c - \mu_i^\mathsf{T}) \Lambda_i^{-1} (P_c^\mathsf{T} V^\mathsf{T} x - \mu_i)} \right) \tag{7.3a}$$

$$- \sum_{x \in X} \log_2 \left(\frac{1}{\sqrt{(2\pi)^{d-m} |\Lambda|}} \mathrm{e}^{-\frac{1}{2}(x^\mathsf{T} V P_n - \mu^\mathsf{T}) \Lambda^{-1} (P_n^\mathsf{T} V^\mathsf{T} x - \mu)} \right) \tag{7.3b}$$

where for $1 \le i \le k$, μ_i is the centroid of cluster C_i in the clustered space and μ is the center of all data objects in the noise space:

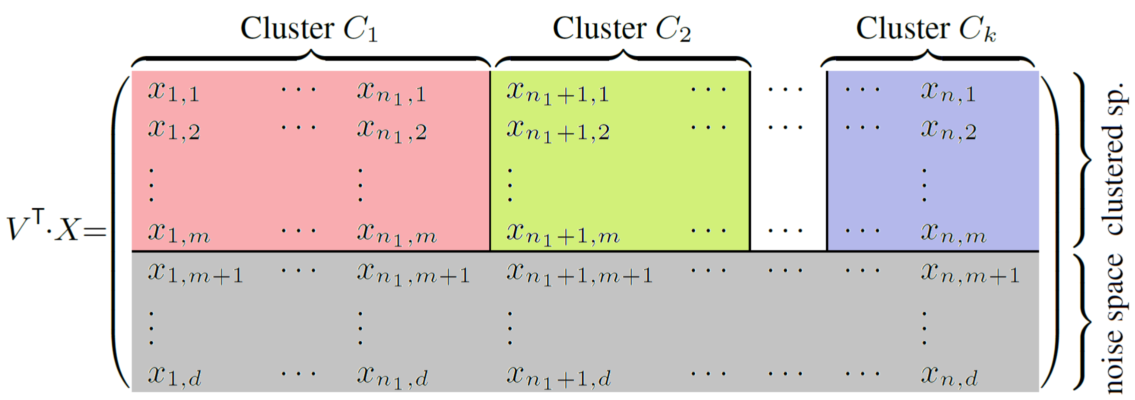
$$\mu_i = \frac{1}{n_i} \sum_{x \in C_i} P_c^\mathsf{T} V^\mathsf{T} x \ \in \mathbb{R}^m \qquad \mu = \frac{1}{n} \sum_{x \in X} P_n^\mathsf{T} V^\mathsf{T} x \ \in \mathbb{R}^{d-m} \tag{7.4}$$

$\Lambda_1, \cdots, \Lambda_k$ are the $(m \times m)$ covariance matrices of the clusters in the m attributes of the clustered subspace and Λ is the overall covariance matrix in the attributes of the noise space:

$$\Lambda_i = \frac{1}{n_i} \sum_{x \in C_i} (P_c^\mathsf{T} V^\mathsf{T} x - \mu_i) \cdot (x^\mathsf{T} V P_c - \mu_i^\mathsf{T}) \quad \in \mathbb{R}^{m \times m} \tag{7.5}$$

$$\Lambda = \frac{1}{n} \sum_{x \in X} (P_n^\mathsf{T} V^\mathsf{T} x - \mu) \cdot (x^\mathsf{T} V P_n - \mu^\mathsf{T}) \quad \in \mathbb{R}^{(d-m) \times (d-m)} \tag{7.6}$$

In this definition, the first line of f (Equation 7.3) corresponds to the maximization of the cluster structure. The second line corresponds to the minimization of the cluster projection in the remaining $(d - m)$ noise dimensions. Note that only one rigid transformation V exists for all k clusters in the clustered subspace and for the noise subspace. The first line corresponds to a number k of Gaussian distributions in the clustered subspace, each defined by the center μ_i and an $m \times m$ covariance matrix Λ_i and weight $\frac{n_i}{n}$. This means that our cluster notion exactly corresponds to that of EM-clustering (with crisp cluster assignment) in the clustered subspace. The noise in the second line is modeled by a single



$$V^{\mathsf{T}} \cdot X = \begin{pmatrix} x_{1,1} & \cdots & x_{n_1,1} & x_{n_1+1,1} & \cdots & \cdots & \cdots & x_{n,1} \\ x_{1,2} & \cdots & x_{n_1,2} & x_{n_1+1,2} & \cdots & \cdots & \cdots & x_{n,2} \\ \vdots & & \vdots & \vdots & & & & \vdots \\ x_{1,m} & \cdots & x_{n_1,m} & x_{n_1+1,m} & \cdots & \cdots & \cdots & x_{n,m} \\ x_{1,m+1} & \cdots & x_{n_1,m+1} & x_{n_1+1,m+1} & \cdots & \cdots & \cdots & x_{n,m+1} \\ \vdots & & \vdots & \vdots & & & & \vdots \\ x_{1,d} & \cdots & x_{n_1,d} & x_{n_1+1,d} & \cdots & \cdots & \cdots & x_{n,d} \end{pmatrix}$$

Figure 7.2.: **Transformation of the data matrix** X (containing data objects as column vectors, grouped by cluster membership) using optimal rigid transform V.

Gaussian distribution function with center μ and covariance matrix Λ in the noise subspace. The FOSSCLU function f is defined as the *negative log-likelihood* (NLLH) of these Gaussian distributions. The NLLH measures the number of bits that are needed to encode the data matrix X by a lossless data compression using our specific cluster model. We will come back to the topic of data compression and fully automatic parameter selection in Section 7.2.4.

The result of FOSSCLU, the transformed data matrix $V^{\mathsf{T}}X$, can also be visualized in Figure 7.2 where we see, that clusters are separated in dimensions 1 through m after applying the transform V to the whole data matrix X. Note that none of the previous methods of subspace, projected and correlation clustering enables such a well-ordered structure, because each cluster has its individual subspace. So the red, green, blue, etc. part is distributed over different dimension patterns. Figure 7.1d demonstrates the great advantage of our method: the resulting matrix can be visualized with scatter plots (7.1d, upper part), with color-coded matrix images (7.1d, lower part), and more sophisticated visualization techniques.

The task of our method FOSSCLU is to optimize simultaneously both the cluster assignment as well as the optimal rigid transform V to optimize the function f given in Definition 7.1.

We can easily see that $\mu_i, \mu, \Lambda_i, \Lambda$ can also be determined from the corresponding center and covariance matrix of the original space:

$$\mu_i = P_c^\mathsf{T} V^\mathsf{T} \cdot \frac{1}{n_i} \sum_{x \in C_i} x \qquad \mu = P_n^\mathsf{T} V^\mathsf{T} \cdot \frac{1}{n} \sum_{x \in X} x \tag{7.7}$$

$$\Lambda_i = P_c^\mathsf{T} V^\mathsf{T} \Sigma_i V P_c \qquad \Lambda = P_n^\mathsf{T} V^\mathsf{T} \Sigma V P_n \tag{7.8}$$

where Σ_i are the $d \times d$ covariance matrices (in original space) of cluster C_i and Σ that of the whole dataset.

7.2.2. ORT: The Optimal Rigid Transform

With usual eigenvalue decomposition (EVD), we can either perform a rigid transformation (rotation) of the data space that maximizes the variance of all data (like principal component analysis (PCA) or singular value decomposition do), or one that minimizes the cluster-specific variance (like ORCLUS, 4C, and other subspace clustering techniques do). As we have already seen in the simple 4D example of Figure 7.1, neither a global eigenvalue decomposition nor a cluster-wise eigenvalue decomposition is sufficient to separate the 2D cluster space from the 2D noise space. Therefore, we need a new, special eigenvalue decomposition, which we call optimal rigid transform (ORT). We show in Section 7.2.5 that our proposed method ORT indeed is optimal for our cluster model of crisp EM clustering in the clustered subspace, as defined in our FOSSCLU function f (cf. Definition 7.1).

Before we can see the elegance of our approach, we have to work ourselves through a fairly lengthy and complex mathematical derivation: the normal eigenvalue decomposition of a (symmetric, positive definite) matrix Σ (the covariance matrix of our data matrix X) *diagonalizes* the matrix by a rotation V_{EVD} of the data space. If we multiply each object $V_{\text{EVD}}^\mathsf{T} \cdot x$, the resulting covariance matrix becomes a diagonal matrix Λ_{EVD}, and we call its elements the *eigenvalues.* The orthonormal (=rotation) matrix V_{EVD} contains the *eigenvectors*. The relationship between eigenvalues and eigenvectors is:

$$\Lambda_{\text{EVD}} = V_{\text{EVD}}^\mathsf{T} \cdot \Sigma \cdot V_{\text{EVD}} \tag{7.9}$$

The specific advantage of the eigenvalue decomposition is, that the data matrix is *de-correlated* by it (when applying $V_{\text{EVD}}^{\mathsf{T}} \cdot X$). The data distribution can then be modeled by a product of univariate distributions rather than by a single multivariate distribution with covariance matrix, at no loss of precision. In addition, among all rigid transforms, eigenvalue decomposition rotates our data matrix such that the product of all univariate variances is minimized.

In contrast to eigenvalue decomposition, ORT does not simply decompose a single matrix. ORT rather decomposes all the covariance matrices (corresponding to the single clusters) and the overall covariance matrix (corresponding to the noise part) simultaneously using the same matrix V:

$$\Lambda_i = P_c^{\mathsf{T}} \cdot V^{\mathsf{T}} \cdot \Sigma_i \cdot V \cdot P_c \quad \text{for all } i \text{ with } 1 \leq i \leq k \tag{7.10}$$

$$\Lambda = P_n^{\mathsf{T}} \cdot V^{\mathsf{T}} \cdot \Sigma \cdot V \cdot P_n \tag{7.11}$$

Usually the resulting matrix V does not produce a *diagonal matrix for each of the matrices* $\Sigma_1, \ldots, \Sigma_k$ and Σ. However, we require that, like in usual eigenvalue decomposition, the product of all variances in $\Lambda_1, \ldots, \Lambda_k$ and Λ is minimized. Since in the clustered subspace each cluster is modeled again by its multivariate Gaussian distribution (using a covariance matrix), the *overall variance* of a cluster is defined by the determinant of this covariance matrix. Thus, the overall optimization goal for ORT is given in the following definition:

Definition 7.2: Optimal Rigid Transform ORT

Given a set of k cluster-specific covariance matrices $\Sigma_1, \ldots, \Sigma_k$ and the overall covariance matrix Σ of X in original data space. Find an orthonormal $d \times d$ matrix V which minimizes the following objective g (called ORT function):

$$g = |P_n^{\mathsf{T}} V^{\mathsf{T}} \Sigma V P_n| \cdot \prod_{1 \leq i \leq k} |P_c^{\mathsf{T}} V^{\mathsf{T}} \Sigma_i V P_c|^{\frac{n_i}{n}} \tag{7.12}$$

where $|\cdots|$ denotes the determinant.

In Section 7.2.5 we prove that ORT truly optimizes f from Definition 7.1, and the determinants are essential for this proof. Intuitively, the determinants represent the overall variances of the clusters in the clustered space ($|P_c^{\mathsf{T}} V^{\mathsf{T}} \Sigma_i V P_c|$, weighted by $\frac{n_i}{n}$), and of the noise in the noise space ($|P_n^{\mathsf{T}} V^{\mathsf{T}} \Sigma V P_n|$). Thus the minimization of the product of these determinants rewards cluster quality in the clustered space and punishes cluster structure in the noise space. A commonly used algorithm for the normal eigenvalue decomposition is the Jacobi transform. The Jacobi transform is an iterative algorithm which *diagonalizes* a matrix by applying orthonormal operations called Givens rotations to it. Each Givens rotation optimizes a 2D plane (p, q), where p and q iterate over all dimension pairs $(1 \leq p < q \leq d)$. The Givens rotations are repeated until convergence. The product of all applied Givens rotations corresponds to the eigenvectors, and the diagonal matrix contains the eigenvalues.

We adopt the general scheme of this algorithm modifying two aspects: Firstly, since we are only interested in separating the clustered subspace from the noise subspace, we can safely restrict ourselves to the cases where $1 \leq p \leq m$ and $m < q \leq d$. So from now on, we always assume that p is in the clustered subspace and q is in the noise subspace. Secondly, we derive an optimization goal for the rotation in (p, q) from the overall goal in Definition 7.2.

The Givens rotation $G_{p,q}(\theta)$ in plane (p, q) with angle θ is an orthonormal $d \times d$ matrix that deviates from identity matrix by the following exceptions: element $[p, p]$ = element $[q, q] = \cos\theta$, element $[p, q] = -\sin\theta$, and element $[q, p] = \sin\theta$:

$$G_{p,q}(\theta) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \cos\theta & & -\sin\theta & & \\ & & & \ddots & & & \\ & & \sin\theta & & \cos\theta & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \tag{7.13}$$

The Givens rotation modifies the ORT matrix V as well as the covariance matrices in the following way:

$$V^{\text{new}} = G_{p,q}(\theta) \cdot V^{\text{old}} \tag{7.14}$$
$$\Sigma^{\text{new}} = G_{p,q}(\theta) \cdot \Sigma^{\text{old}} \cdot G_{p,q}(\theta)^{\mathsf{T}} \tag{7.15}$$
$$\Sigma_i^{\text{new}} = G_{p,q}(\theta) \cdot \Sigma_i^{\text{old}} \cdot G_{p,q}(\theta)^{\mathsf{T}} \tag{7.16}$$

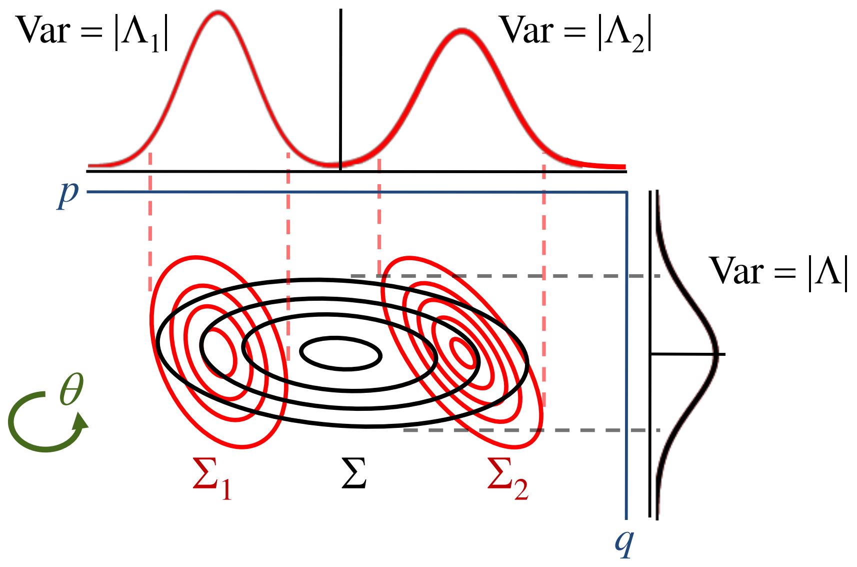


Figure 7.3.: **Example of plane rotation for two clusters**. The Givens rotation $G_{p,q}(\theta)$ is applied to plane (p, q) by angle θ. Rotating the plane corresponds to rotating the covariance matrices. It represents a trade-off for minimizing the variance of each cluster in the clustered space and the total variance in the noise space. Hence, the product $|\Lambda_1|^{n_1/n} \cdot |\Lambda_2|^{n_2/n} \cdot |\Lambda|$ is minimized.

So the sequence of Givens rotations is multiplied with the matrix V from the left (initialized by the identity matrix) and from both sides to Σ and Σ_i (both initialized by the true covariance matrices). So at convergence in step ℓ we will have:

$$V^{\text{final}} = G^{(\ell)}_{m,d} \cdot \ldots \cdot G^{(\ell)}_{1,m+1} \cdot \ldots \cdot G^{(1)}_{m,d} \cdot \ldots \cdot G^{(1)}_{1,m+1} \cdot I \tag{7.17}$$

and analogously for Σ and Σ_i. The final covariance matrices in clustered and noise space are determined by $\Lambda_i^{\text{final}} = P_c^{\mathsf{T}} \Sigma_i^{\text{final}} P_c$.

To be able to select the rotation angle θ in an optimal way, we have to find out how a Givens rotation affects our objective function g from Definition 7.2 (the product of determinants, cf. Figure 7.3). We define the function $g_{p,q}(\theta)$ which is our ORT function g with a focus on the current optimization step for a given plane (p, q) and the rotation angle θ to be optimized:

$$g_{p,q}(\theta) = |P_n^\mathsf{T} G_{p,q}(\theta)\Sigma^{\text{old}} G_{p,q}^\mathsf{T}(\theta) P_n| \cdot \prod_{1\le i\le k} |P_c^\mathsf{T} G_{p,q}(\theta)\Sigma_i^{\text{old}} G_{p,q}^\mathsf{T}(\theta) P_c| \tag{7.18}$$

The product $P_c^\mathsf{T} \cdot G_{p,q}(\theta)$ has the following form:

$$P_c^\mathsf{T} G_{p,q}(\theta) = \begin{pmatrix} 1 & & \cdots & & 0 & 0 & \cdots & 0 \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ \vdots & & & \cos\theta & \vdots & \vdots & -\sin\theta & \vdots \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ 0 & & \cdots & & 1 & 0 & \cdots & 0 \end{pmatrix} \tag{7.19}$$

It deviates from P_c^T by just two entries: element $[p, p]$ is $\cos\theta$ (instead of 1) and element $[p, q]$ is $-\sin\theta$ (instead of 0). This means multiplying it from left and right modifies our covariance matrix Σ_i by multiplying row and column p with $\cos\theta$ and subtracting row/column q from it (with weight $\sin\theta$). Likewise, our overall covariance matrix Σ is modified by multiplying row and column q with $\cos\theta$ and adding $\sin\theta$ times the row and column p to it. We exploit the multi-linearity of the determinant in rows and columns and in this way extract $\sin\theta$ and $\cos\theta$ from the determinant:

$$\begin{aligned} g_{p,q}(\theta) = \quad & |P_n^\mathsf{T} G_{p,q}(\theta)\Sigma^{\text{old}} G_{p,q}^\mathsf{T}(\theta) P_n| \cdot \prod_{1\le i\le k} |P_c^\mathsf{T} G_{p,q}(\theta)\Sigma_i^{\text{old}} G_{p,q}^\mathsf{T}(\theta) P_c| \\ = \quad & \Big(\sin^2\theta \cdot |P_n^\mathsf{T} S_{p,q}\Sigma^{\text{old}} S_{p,q} P_n| + 2\sin\theta\cos\theta \cdot |P_n^\mathsf{T}\Sigma^{\text{old}} S_{p,q} P_n| \\ & + \cos^2\theta \cdot |P_n^\mathsf{T}\Sigma^{\text{old}} P_n|\Big) \qquad (7.20) \\ \cdot \prod_{1\le i\le k} & \Big(\sin^2\theta \cdot |P_c^\mathsf{T} S_{p,q}\Sigma_i^{\text{old}} S_{p,q} P_c| - 2\sin\theta\cos\theta \cdot |P_c^\mathsf{T}\Sigma_i^{\text{old}} S_{p,q} P_c| \\ & + \cos^2\theta \cdot |P_c^\mathsf{T}\Sigma_i^{\text{old}} P_c|\Big)^{n_i/n} \end{aligned}$$

where $S_{p,q}$ is the swapping matrix which exchanges the rows p and q of an arbitrary matrix M when multiplied from the left side to M, and the corresponding columns when multiplied on the right side of M. $S_{p,q}$ looks like the $d \times d$

identity matrix with the exceptions that element $[p, p]$ = element $[q, q] = 0$ and element $[p, q]$ = element $[q, p] = 1$:

$$S_{p,q} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & \cdots & 1 & & \\ & & \vdots & \ddots & \vdots & & \\ & & 1 & \cdots & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \tag{7.21}$$

We define the following substitutions: $n_0 := n$, and for all clusters C_i with $1 \leq i \leq k$:

$$a_0 := |P_n^\mathsf{T} S_{p,q} \Sigma^{\text{old}} S_{p,q} P_n| \qquad a_i := |P_c^\mathsf{T} S_{p,q} \Sigma_i^{\text{old}} S_{p,q} P_c| \tag{7.22}$$

$$b_0 := |P_n^\mathsf{T} \Sigma^{\text{old}} S_{p,q} P_n| \qquad b_i := -|P_c^\mathsf{T} \Sigma_i^{\text{old}} S_{p,q} P_c| \tag{7.23}$$

$$c_0 := |P_n^\mathsf{T} \Sigma^{\text{old}} P_n| \qquad c_i := |P_c^\mathsf{T} \Sigma_i^{\text{old}} P_c| \tag{7.24}$$

This enables us to write $g_{p,q}(\theta)$ in an elegant way in which we treat the noise space like a cluster $C_{i=0}$:

$$g_{p,q}(\theta) = \prod_{0 \leq i \leq k} (a_i \sin^2\theta + 2b_i \sin\theta\cos\theta + c_i \cos^2\theta)^{n_i/n} \tag{7.25}$$

Now we minimize $g_{p,q}(\theta)$ by setting its derivative to zero. Applying product and chain rule gives us:

$$\begin{aligned} \frac{dg_{p,q}(\theta)}{d\theta} = \sum_{0 \leq i \leq k} \Bigg(& \frac{n_i}{n} \Big(a_i \sin^2\theta + 2b_i \sin\theta\cos\theta + c_i \cos^2\theta \Big)^{\frac{n_i}{n} - 1} \\ & \cdot 2 \Big(-b_i \sin^2\theta + (a_i - c_i)\sin\theta\cos\theta + b_i \cos^2\theta \Big) \\ & \cdot \prod_{\substack{0 \leq j \leq k \\ j \neq i}} \Big(a_j \sin^2\theta + 2b_j \sin\theta\cos\theta + c_j \cos^2\theta \Big)^{\frac{n_j}{n}} \Bigg) \end{aligned} \tag{7.26}$$

The roots of the derivative can be found by factorizing out the terms $(a_i \sin^2\theta + 2b_i \sin\theta\cos\theta + c_i \cos^2\theta)$, to the power of $(\frac{n_i}{n} - 1)$. After this step one of these terms is completely drawn out of the sum, and the rest remains inside but only to the power of 1. Additionally, for each of the $k+1$ factors, we factor out $\cos^2\theta$ to change $\sin^2\theta$ into $\tan^2\theta$, $\sin\theta\cos\theta$ into $\tan\theta$, and $\cos^2\theta$ into 1:

$$= \quad \Bigg(\prod_{0\le i\le k} \Big(a_i \sin^2\theta + 2b_i \sin\theta\cos\theta + c_i\cos^2\theta \Big)^{\frac{n_i}{n}-1} \Bigg) \cdot \tfrac{2}{n}\cos^{2k+2}\theta \tag{7.27a}$$

$$\cdot \sum_{0\le i\le k} \Bigg(\Big(-b_i n_i \tan^2\theta + (a_i - c_i) n_i \tan\theta + b_i n_i \Big) \tag{7.27b}$$

$$\cdot \prod_{0\le j<i} (a_j \tan^2\theta + 2b_j\tan\theta + c_j) \tag{7.27c}$$

$$\cdot \prod_{i<j\le k} (a_j \tan^2\theta + 2b_j\tan\theta + c_j) \Bigg) \tag{7.27d}$$

By substituting $\xi := \tan\theta$ we can easily see the equivalence with a polynomial of degree $2k+2$ which is the sum of $k+1$ polynomials (of the same degree). These $k+1$ polynomials differ only in one factor (Equation 7.27b), resulting from the chain rule, which replaces the i-th factor in Equation 7.27c or 7.27d. An efficient way to determine the coefficients of the final polynomial is first to pre-compute the coefficients of Equation 7.27c for all $1 \le i < k$ by multiplying the single factor-polynomials one by one, and memoizing the intermediate results in $k-1$ arrays. Then, the polynomials of Equation 7.27d are determined in the same way but in reverse order $k \ge i > 1$, multiplied with the corresponding memoized intermediate result of Equation 7.27c and the remaining term (Equation 7.27b), and added to the coefficients of the resultant polynomial. This requires a linear number of $4k$ polynomial multiplications. Finally, we determine the $2k+2$ roots $\xi_1, \ldots, \xi_{2k+2}$ of the resultant polynomial (e.g., by an eigenvalue decomposition of the companion matrix). We back-substitute $\theta_i = \arctan\xi_i$ of each non-complex root of the polynomial to find the global minimum

$$\theta_{\min} = \operatorname*{arg\,min}_{1\le i\le 2k+2} g_{p,q}(\theta_i). \tag{7.28}$$

Once we have found $\theta_{\min}$, we can apply the Givens rotation by setting

$$V^{\text{new}} := G_{p,q}(\theta_{\min}) \cdot V^{\text{old}} \tag{7.29}$$

and applying it likewise to Σ and Σ_i. The determination of $\theta_{\min}$ and its application in a Givens rotation is repeated for all planes $(p, q) \in (\{1, ..., m\} \times \{m + 1, ..., d\})$, and this is repeated until convergence. Finally, we have the ORT V, and the covariance matrices in clustered and noise space can be determined by applying the corresponding projection matrices to the final versions of Σ and $\Sigma_1, ..., \Sigma_k$.

7.2.3. The Algorithm FOSSCLU

In our alternating least squares scheme, we are performing the two steps cluster assignment and subspace determination. Both steps are optimizing the objective function f given in Definition 7.1, as we show in Section 7.2.5. For the cluster assignment step, we assume that the orthonormal matrix V^T is fixed. We perform a crisp EM-clustering in the clustered subspace, i.e. each point is assigned to that cluster which maximizes the point's likelihood, according to the weighted Gaussian probability density function (defined by the weight $\frac{n_i}{n}$, center μ_i, and covariance matrix Σ_i of the cluster):

$$i = \arg\max_{1 \le i \le k} \frac{\frac{n_i}{n}}{\sqrt{(2\pi)^m |\Lambda_i|}} \mathrm{e}^{-\frac{1}{2}(x^\mathsf{T} V P_c - \mu_i^\mathsf{T}) \Lambda_i^{-1} (P_c^\mathsf{T} V^\mathsf{T} x - \mu_i)} \tag{7.30}$$

Algorithm 7.1 summarizes FOSSCLU in pseudo-code[1]. We initialize the FOSSCLU cluster centers by randomly sampling points from the data set. The iterative algorithmic scheme then starts with determining the optimal rigid transform. Subsequently, the points are re-assigned as described above. Both steps are iterated until convergence. The cluster weight $\frac{n_i}{n}$ is not considered in the first five iterations of the algorithm to prevent the initial clusters from running empty.

7.2.4. Minimum Description Length (MDL)

The idea of MDL (cf. Section 2.2.1) is to use the result of an arbitrary data mining algorithm as a statistical model which can be exploited for the efficient compression of the dataset. The better the model fits to the data, the higher

[1]Our implementation for FOSSCLU is available at `https://s.goebl.net/research`.

Algorithm 7.1: Algorithm FOSSCLU

```
Input: matrix $X$, int $k$, int $m$
Output: $\{C_1, \ldots, C_k\}$, $V^\mathsf{T}$
  // Initialization
  $\hat{\mu} \longleftarrow \frac{1}{n} \sum_{x \in X} x$
  $\Sigma \longleftarrow \frac{1}{n} \sum_{x \in X} (x - \hat{\mu}) \cdot (x - \hat{\mu})^\mathsf{T}$
  for $i \longleftarrow 1$ to $k$ do
      $\mu_i \longleftarrow$ random point from $X$
  end for
  forall $x \in X$ do
      $x$.clusterID $\longleftarrow \arg\min_{1 \le i \le k} (x - \hat{\mu}_i)^2$
  end forall
  repeat
      // Update cluster models
      for $i \longleftarrow 1$ to $k$ do
          $\hat{\mu}_i \longleftarrow \frac{1}{n} \sum_{x \in C_i} x$
          $\Sigma_i \longleftarrow \frac{1}{n} \sum_{x \in C_i} (x - \hat{\mu}_i) \cdot (x - \hat{\mu}_i)^\mathsf{T}$
          $C_i \longleftarrow \emptyset$
      end for
      $V^\mathsf{T} \longleftarrow$ ORT$(\Sigma, \{\Sigma_1, \ldots, \Sigma_k\}, m)$            ▷ Algorithm 7.2
      for $i \longleftarrow 1$ to $k$ do
          $\mu_i \longleftarrow V^\mathsf{T} \hat{\mu}_i$
          $\Lambda_i \longleftarrow V^\mathsf{T} \Sigma_i V$
      end for
      // Re-cluster the objects
      forall $x \in X$ do
          $j \longleftarrow \arg\max_{1 \le i \le k} \frac{\frac{n_i}{n}}{\sqrt{|\Lambda_i|}} e^{-\frac{1}{2}(x^\mathsf{T} V P_c - \mu_i^\mathsf{T}) \Lambda_i^{-1} (P_c^\mathsf{T} V^\mathsf{T} x - \mu_i)}$
          $C_j \longleftarrow C_j \cup \{x\}$
      end forall
  until convergence
```

Algorithm 7.2: Algorithm ORT

```
Input: matrix $\Sigma$, matrix $\{\Sigma_1, ...\Sigma_k\}$, int $m$
Output: matrix $V$
  // This version extends Jacobi-diagonalization [Pre+07]
1 $V^\top \longleftarrow (d \times d)$ identity matrix
2 repeat
3     for $i \longleftarrow 1$ to $m$ do
4         for $j \longleftarrow m+1$ to $d$ do
5             $\theta \longleftarrow$ optimal rotation angle (c.f. solution of Section 7.2.2)
6             $G \longleftarrow (d \times d)$ identity matrix
7             $g_{i,i} \longleftarrow \cos\theta$
8             $g_{j,j} \longleftarrow \cos\theta$
9             $g_{i,j} \longleftarrow \sin\theta$
10            $g_{j,i} \longleftarrow -\sin\theta$
11            $\Sigma \longleftarrow G \cdot \Sigma \cdot G^\top$                          }
12            $\Sigma_{1,...,k} \longleftarrow G \cdot \Sigma_{1,...,k} \cdot G^\top$        } possible in $O(d)$ time!
13            $V \longleftarrow G \cdot V$                                                   }
14        end for
15    end for
16 until convergence
```

the compression factor. Therefore, MDL can be used as a selection criterion among different models or it can be directly applied in the optimization step of the data mining algorithm, like we do in FOSSCLU. To avoid overfitting and overly complex models, MDL techniques add the amount of information which is needed to code the parameters of the model (called code-book) to the code-length of the compressed data. We define in the following how parameters of our method, the results, and the actual data can be efficiently coded.

Definition 7.3: MDL coding of FOSSCLU

Let $X \in \mathbb{R}^{d \times n}$ be the data matrix, $k \in \mathbb{N}$ the number of clusters $(1 \leq k \leq n)$, $m \in \mathbb{N}$ the dimensionality of the clustered subspace $(1 \leq m \leq d)$, V an orthonormal $d \times d$ matrix (optimal rigid transform), $\mu_1, ..., \mu_k \in \mathbb{R}^m, \mu \in \mathbb{R}^{d-m}$ the centroids in the clustered and noise subspace, respectively, and $\Lambda_1, \ldots, \Lambda_d$ and Λ the corresponding covariance matrices. The MDL coding of the dataset consists of the binary codes:

(1) k and m using $\log_2 n$ and $\log_2 d$ bits, respectively, and V using $\frac{d \cdot (d-1)}{4} \cdot \log_2 n$ bits,

(2) $\mu_1, ..., \mu_k, \Lambda_1, ...\Lambda_k, \mu$, and Λ using $\left(\left(\frac{3}{4} + \frac{1}{4} \cdot m\right) mk + \frac{3}{4}(d-m) + \frac{1}{4}(d-m)^2\right) \log_2 n$ bits,

(3) the data objects using f bits, where f is the FOSSCLU function (i.e. the negative log-likelihood of all data under our cluster model).

The data objects are coded with a number of bits equal to the negative log-likelihood of the probability density function which is associated to the cluster representative. The negative log-likelihood of all data objects is exactly the FOSSCLU function f. Note that the cluster assignment information (requiring $-\log_2 \frac{n_i}{n}$ bits per object) is already contained in f. We call element (1) the constant code book (since the number of bits is constant during the whole optimization process), element (2) the variable code book, and element (3) the log-likelihood cost.

Our MDL representation intentionally does not specify any numerical precision to which the data matrix is coded (after clustering and application of the optimal subspace transformation). It can be shown that the choice of the grid resolution has no influence on the assessment of the goodness of the model [BFP08]. The parameters of the code book like, e.g., our optimal rigid transform V are taken into account with $\frac{1}{2}\log_2 n$ bits each. The length of the MDL coding given in Definition 7.3 can in particular be used to find optimal parameter settings for k and m. FOSSCLU can automatically select that parameter setting which is optimal for data compression to avoid over- and under-fitting. However, the user is also free to set m and k according to their requirements (e.g., to set $m = 2$ or $m = 3$ for visualization in scatter plots).

7.2.5. Correctness and Convergence

We prove the convergence of FOSSCLU by showing that both the cluster assignment as well as the optimal rigid transform optimize the same objective function given in Definition 7.1.

Lemma 7.1. *The assignment step provided in Section 7.2.3 optimizes the objective function f from Definition 7.1.*

Proof. In the cluster assignment step, V, μ, μ_i, Λ, and Λ_i with $(1 \leq i \leq k)$ are fixed. The second line of f (Equation 7.3b: noise space) is constant for all clusters C_i and is not changed by the cluster assignment. Leaving out all those values in the FOSSCLU function f which are *constant in the assignment step* yields:

$$f = -\sum_{1\leq i\leq k}\sum_{x\in C_i}\log_2\left(\frac{\frac{n_i}{n}}{\sqrt{(2\pi)^m|\Lambda_i|}}\mathrm{e}^{-\frac{1}{2}(x^\mathsf{T}VP_c-\mu_i^\mathsf{T})\Lambda_i^{-1}(P_c^\mathsf{T}V^\mathsf{T}x-\mu_i)}\right) + O(\text{const.}) \tag{7.31}$$

This is exactly the logarithm of the cluster assignment function provided in Section 7.2.2, the cluster assignment function of crisp EM clustering in the clustered subspace. Thus, the minimization of the cluster assignment function results in the minimization of the FOSSCLU function. □

Lemma 7.2. *The optimal rigid transform V optimizes the objective function f from Definition 7.1.*

Proof. First, we show that the term h which is part of the second line of f (Equation 7.3b) corresponding to the noise space is constant:

$$h := -\sum_{x\in X} \log_2 \mathrm{e}^{-\frac{1}{2}(x^\mathsf{T} V P_n - \mu^\mathsf{T})\Lambda^{-1}(P_n^\mathsf{T} V^\mathsf{T} x - \mu)} \tag{7.32}$$

Let $\Lambda = WDW^\mathsf{T}$ be the eigenvalue decomposition of Λ. Then the entries of D contain the variances of $W^\mathsf{T} P_n^\mathsf{T} V^\mathsf{T} x$, and the center of all data $x \in X$ in this space is $W^\mathsf{T} \cdot \mu$. Then we can re-write h using the j-th unit vector e_j as follows:

$$\begin{aligned} h &= \frac{\log_2(\mathrm{e})}{2} \sum_{x\in X} \sum_{1\le j\le d-m} \frac{(e_j W^\mathsf{T} P_n^\mathsf{T} V^{\mathsf{P}\mathsf{T}} x - e_j W^\mathsf{T} \mu)^2}{D_{j,j}} \\ &= \sum_{1\le j\le d-m} \frac{\log_2(\mathrm{e}) \cdot n}{2 \cdot D_{j,j}} \cdot \underbrace{\tfrac{1}{n} \sum_{x\in X} (e_j W^\mathsf{T} P_n^\mathsf{T} V^\mathsf{T} x - e_j W^\mathsf{T} \mu)^2}_{=D_{j,j}} \end{aligned} \tag{7.33}$$

where the sum corresponds to the formula of the variance $D_{j,j}$ of attribute j. We obtain:

$$h = \sum_{1\le j\le d-m} \frac{n \log_2(\mathrm{e})}{2} = \tfrac{1}{2} n (d-m) \log_2(\mathrm{e}) \quad \in O(\text{const.}) \tag{7.34}$$

We obtain an analogous result for each cluster:

$$h_i = -\sum_{x\in C_i} \log_2 \mathrm{e}^{-\frac{1}{2}(x^\mathsf{T} V P_n - \mu_i^\mathsf{T})\Lambda_i^{-1}(P_n^\mathsf{T} V^\mathsf{T} x - \mu_i)} \quad \in O(\text{const.}) \tag{7.35}$$

Now we can separate in the FOSSCLU function all the terms which are constant in the ORT step:

$$\begin{aligned} f &= \sum_{1\le i\le k}\left(h_i - \sum_{x\in C_i}\log_2\Big(\frac{\frac{n_i}{n}}{\sqrt{(2\pi)^m|\Lambda_i|}}\Big)\right) + h - \sum_{x\in X}\log_2\Big(\frac{1}{\sqrt{(2\pi)^m|\Lambda|}}\Big) \\ &= -\Big(\sum_{1\le i\le k}\sum_{x\in C_i}\log_2\frac{1}{\sqrt{|\Lambda_i|}}\Big) - \Big(\sum_{x\in X}\log_2\frac{1}{\sqrt{|\Lambda|}}\Big) + O(\text{const.}) \\ &= \frac{1}{2}\log_2\Big(|\Lambda|\cdot\prod_{1\le i\le k}|\Lambda_i|\Big) + O(\text{const.}) \\ &= \frac{1}{2}\log_2\Big(|P_n^{\mathsf T}V^{\mathsf T}\Sigma VP_n|\cdot\prod_{1\le i\le k}|P_c^{\mathsf T}V^{\mathsf T}\Sigma_i VP_c|\Big) + O(\text{const.}) \end{aligned} \tag{7.36}$$

which is the logarithm of the ORT objective function g (Equation 7.12). Minimization of g in the ORT step leads to minimization of the FOSSCLU function f. □

Together, Lemma 7.1 and 7.2 demonstrate that the two steps, cluster assignment and ORT really fit together, and that ORT is really that transformation of the data space that fits to EM clustering in a common (arbitrarily oriented) subspace. These lemmata are also essential to prove that our alternating least squares algorithm defined in Section 7.2.3 converges:

Lemma 7.3. *The algorithm FOSSCLU (Algorithm 7.1) converges.*

Proof. The function f is monotonically decreasing in each step of the algorithm: the assignment step minimizes f with a fixed ORT and varying cluster assignments. The update step minimizes f with a fixed cluster assignment and varying ORT. Since f is lower bounded the convergence is guaranteed. □

We note without giving a formal proof that MDL defined in Section 7.2.4 is also monotonic with f.

7.2.6. Complexity of FOSSCLU

The complexity of ORT is identical to that of most other well-known eigenvalue decompositions like Jacobi, Housholder, Givens, QR etc. Let ℓ_{ORT} be the number of iterations needed until convergence. The complexity is determined by the innermost loop which performs the Givens rotation. This is done by one multiplication and one addition for $O(d)$ matrix entries. The Givens rotation is performed $\ell_{\text{ORT}} \cdot d^2$ times, resulting in an overall complexity of $\ell_{\text{ORT}} \cdot d^3$. We observed a Jacobi-like iteration count. The root-finding method as described in Section 7.2.2 is performed in the worst case $\ell_{\text{ORT}} \cdot d^2$ times and does not affect the asymptotic complexity.

The complexity of FOSSCLU with fixed k and m equals that of EM clustering since in the update step the eigenvalue decomposition of EM is replaced by ORT with the same complexity. The assignment step also involves the multiplication of each object with an eigenvector matrix. The distance computation of FOSSCLU needs only m operations (compared to d of EM), but this also has no effect on the overall complexity. The number of iterations is also comparable to that of EM clustering. Like k-means and EM-clustering, FOSSCLU may converge to a local minimum. This is handled by multiple random initializations.

7.3. Fuzzy FOSSCLU

Often the EM clustering algorithm is applied with a fuzzy association of objects to clusters, where each point x belongs to every cluster C_i to the degree $w_i(x)$, with

$$\sum_{1 \le i \le k} w_i(x) = 1. \tag{7.37}$$

The degree is proportional to the likelihood of the point in the cluster, and can be determined by Bayes' theorem. This can also be done in our clustered subspace:

$$w_i(x) = \frac{\frac{\frac{\tilde{n}_i}{n}}{\sqrt{(2\pi)^m |\tilde{\Lambda}_i|}} \mathrm{e}^{-\frac{1}{2}(x^\top V P_c - \tilde{\mu}_i^\top)\tilde{\Lambda}_i^{-1}(P_c^\top V^\top x - \tilde{\mu}_i)}}{\sum_{1 \le j \le k} \frac{\frac{\tilde{n}_j}{n}}{\sqrt{(2\pi)^m |\tilde{\Lambda}_j|}} \mathrm{e}^{-\frac{1}{2}(x^\top V P_c - \tilde{\mu}_j^\top)\tilde{\Lambda}_j^{-1}(P_c^\top V^\top x - \tilde{\mu}_j)}} \tag{7.38}$$

Then, the weight of the cluster is the sum of the weights of the assignments,

$$\tilde{n}_i = \sum_{x \in X} w_i(x). \tag{7.39}$$

The center $\tilde{\mu}_i$ and covariance matrix $\tilde{\Lambda}_i$ can be determined likewise, and this is also possible in our clustered subspace:

$$\tilde{\mu}_i = \frac{1}{\tilde{n}_i} \sum_{x \in X} w_i(x) \cdot P_c^\mathsf{T} V^\mathsf{T} x \tag{7.40}$$

$$\tilde{\Lambda}_i = \frac{1}{\tilde{n}_i} \sum_{x \in X} w_i(x)(P_c^\mathsf{T} V^\mathsf{T} x - \tilde{\mu}_i)(x^\mathsf{T} V P_c - \tilde{\mu}_i^\mathsf{T}) \tag{7.41}$$

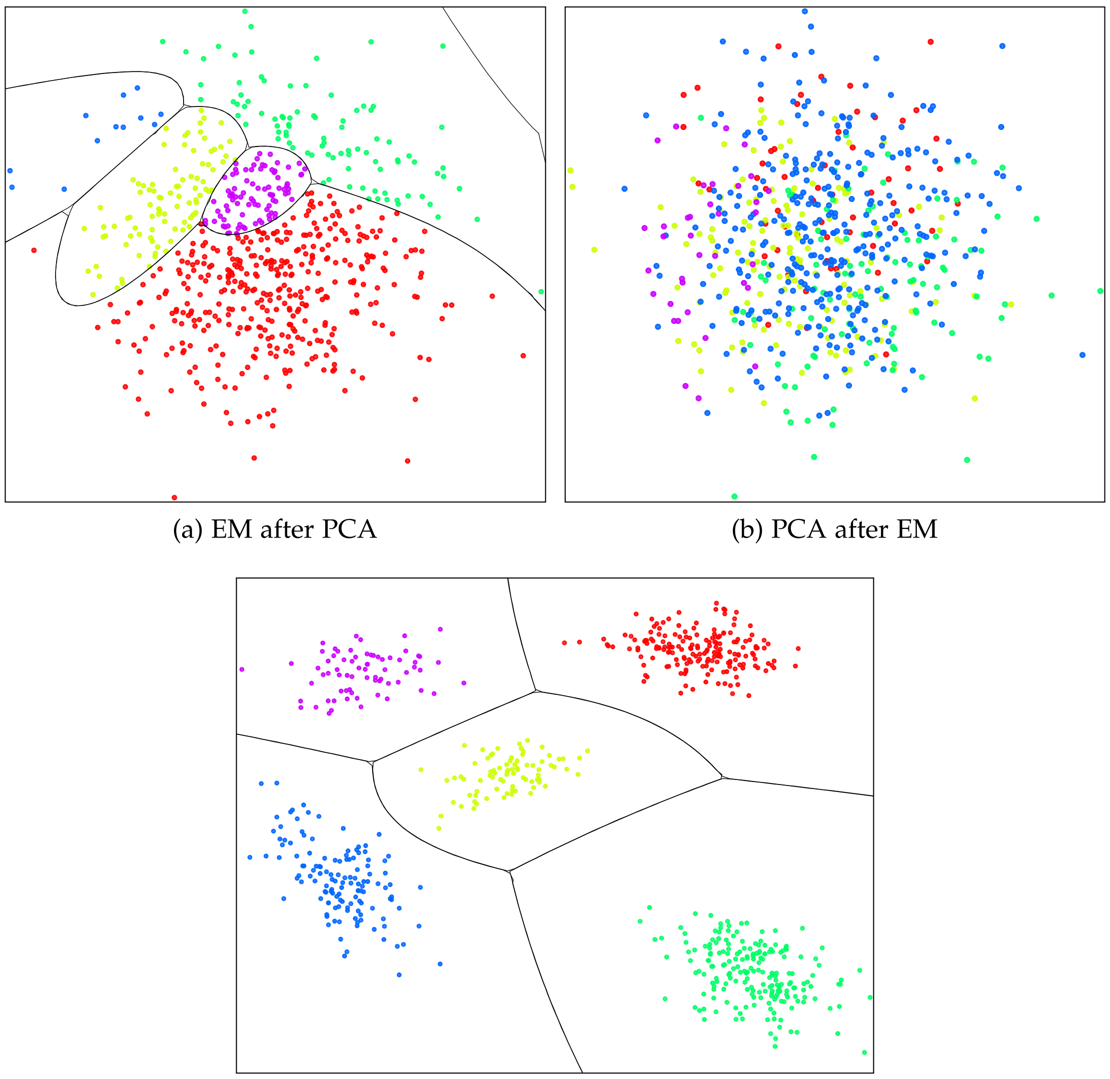
We can determine ORT from the fuzzily determined covariance matrices as well, and the complete FOSSCLU algorithm is adapted in a straightforward way.

7.4. Experiments

In this section, we evaluate (a) the quality of the visualization and (b) the clustering quality of the results gained by FOSSCLU.

For evaluating visualization, we compare FOSSCLU to PCA. It is the most widely used technique for dimensionality reduction and conceptually most related to FOSSCLU. We additionally compare to independent component analysis (ICA; using symmetric approach and nonlinearity tanh) since this technique has recently proven to be useful for cluster identification, see, e.g., [BFP08]. We also compare to the sequential workflows of first performing PCA respective ICA and, then, clustering in the low-dimensional space using EM, call *EM after PCA* and *EM after ICA*, respectively. We further compare to LDA-kMeans [DL07]. This algorithm integrates with linear discriminant analysis a supervised feature transformation technique into clustering, cf. Section 6.1.

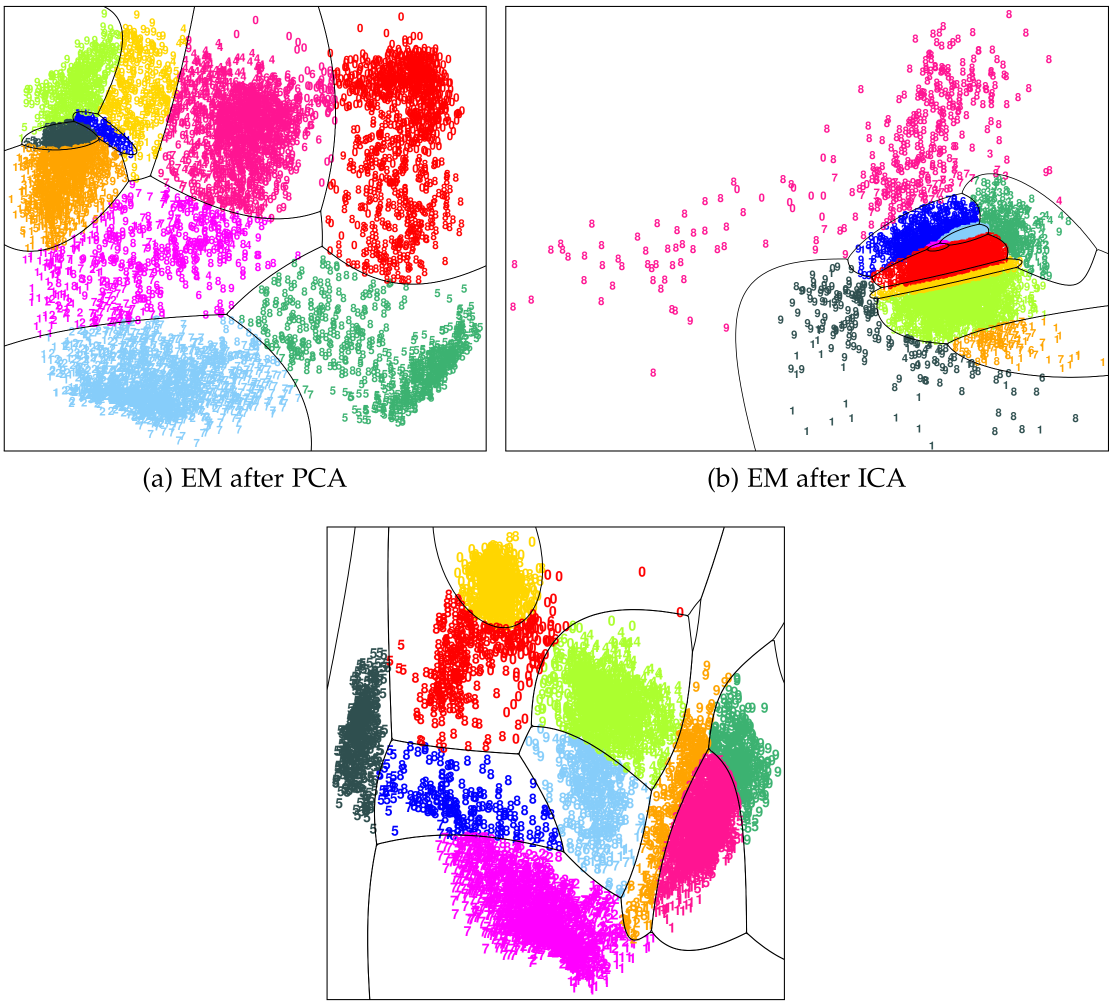
In evaluating cluster quality, we compare to two state-of-the-art approaches to generalized subspace clustering: ORCLUS [AY00] performing iterative partitioning based on *k*-means and the local density-based approach 4C [Böh+04b]. As evaluation measure we report the normalized mutual information (NMI) (cf. Section 2.2.2) between the class labels and the cluster ids generated by the algorithm. The NMI ranges from zero to one with one indicating a perfect clustering.


(a) EM after PCA

(b) PCA after EM

(c) FOSSCLU

Figure 7.4.: Visualization of a synthetic 12D dataset with 5 clusters in 2 clustered and 10 noise dimensions. Only FOSSCLU (c) can identify clusters and subspace.



(a) EM after PCA

(b) EM after ICA

(c) FOSSCLU

Figure 7.5.: **Visualization of PENDIGITS [DG19].** Cluster labels are coded by color, class labels by digits $(0,\ldots,9)$. FOSSCLU (c) yields the highest purity.

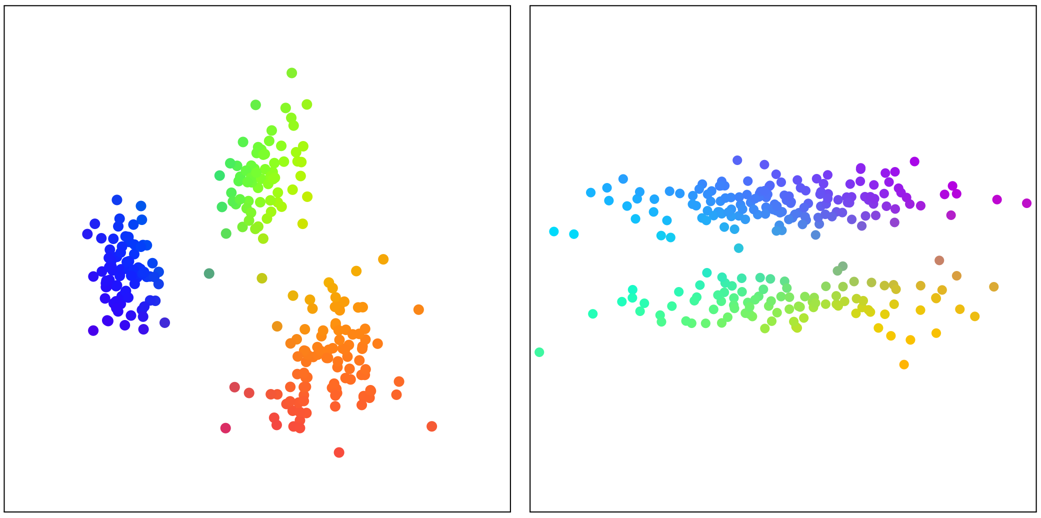

(a) Fuzzy FOSSCLU (b) Fuzzy EM after PCA

Figure 7.6.: Fuzzy association of 3 clusters in 7D with 5D noise. The cluster association is color-coded in RGB (e.g. pure *blue* means 100% association to the blue cluster). Only FOSSCLU (a) correctly identifies clusters and subspace.

7.4.1. Interpretable Visualization

We created a dataset with five Gaussian clusters of varying covariance in 2D space consisting of 70 to 200 points and with ten uni-modal Gaussian noise dimensions. The whole dataset is transformed by an arbitrary 12D rotation matrix. Only FOSSCLU reconstructs the 2D cluster structure perfectly (Figure 7.4). EM after PCA fails to detect the clusters: due to PCA the information relevant for clustering is lost. Performing EM in high-dimensional space yields poor results due to the curse of dimensionality. The results of ICA and EM after ICA are very similar and not depicted here due to space limitations.

The visualization results for the PENDIGITS dataset from the UCI machine learning repository [DG19] for FOSSCLU are displayed in Figure 7.5c. Figures 7.5a and 7.5b draw the results of EM after PCA and EM after ICA. Only FOSSCLU yields a meaningful visualization with multiple pronounced clusters: The leftmost dark gray cluster is well separated from all the others containing to 99.7% the digit 5. The rightmost dark green cluster is also very pure, being 92.2% composed of the digit 9. The uppermost yellow cluster contains 82% of the 0 digits. The neighboring red cluster is composed of to 99.6% of the two similar digits 0 and 8. Hardly any cluster structure can be seen in the EM after PCA result. Considering the cluster content, only the digit 9 is fairly identified since it constitutes 88.2% of the left uppermost light green cluster. For EM after ICA only one digit (in this case the digit 9) is well identified: for 92% it consists of the dark red cluster.

As introduced in Section 7.3, FOSSCLU allows fuzzy association as visualized in Figure 7.6a. In comparison, EM after PCA (Figure 7.6b) is not able to identify the subspace containing the three clusters.

7.4.2. Accurate Clustering

Based on the synthetic dataset displayed in Figure 7.1, we systematically vary two properties: the number of uni-modal dimensions without any cluster information (from 2 to 20), and the variance of the clusters (multiplying the standard deviations of the clusters from Figure 7.4 with factors varying from 0.6 to 2.0. ORCLUS also needs the average subspace dimensionality to be set (in our case

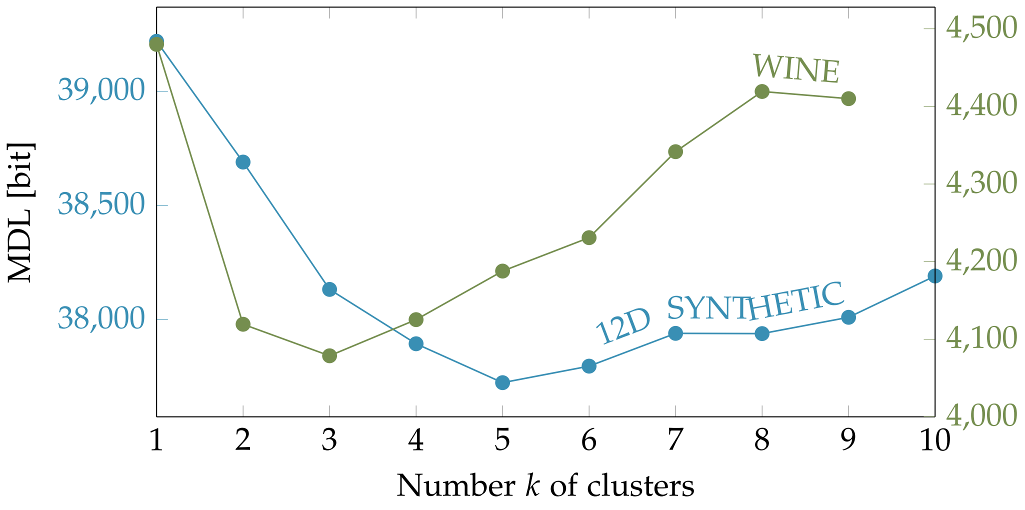
Table 7.2.: Quality (NMI) of results of real-world datasets. FOSSCLU outperforms all other methods significantly ($\alpha < 0.007$ for PENDIGITS/ORCLUS; $\alpha < 0.0002$ for METABOLIC/EM after PCA; all others even better)

	WINE	METABOLIC	PENDIGITS
FOSSCLU	$\mathbf{0.87 \pm 0.04}$	$\mathbf{0.88 \pm 0.01}$	$\mathbf{0.77 \pm 0.03}$
LDA-kMeans	0.45 ± 0.00	0.40 ± 0.00	0.71 ± 0.03
EM after PCA	0.45 ± 0.04	0.84 ± 0.00	0.69 ± 0.03
EM after ICA	0.07 ± 0.05	0.04 ± 0.03	0.63 ± 0.04
ORCLUS	0.07 ± 0.02	0.03 ± 0.02	0.67 ± 0.10
4C	0.35 ± 0.00	0.78 ± 0.07	0.48 ± 0.00

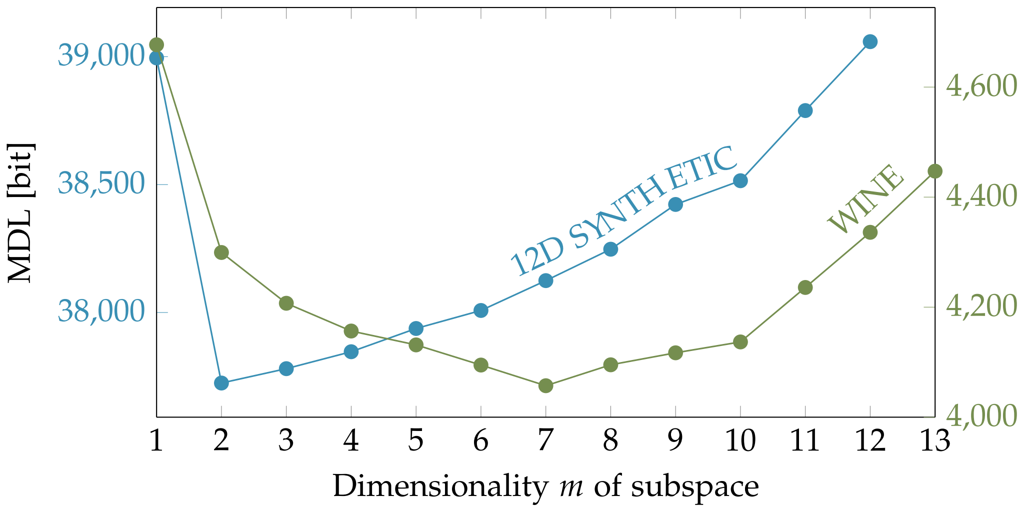
2). The algorithm 4C requires the parameters ϵ and $MinPts$ for its density based clustering and λ for subspace correlation. We run 4C for $\epsilon = 4 \dots 20$, $\lambda = 3 \dots d$ and $MinPt = 1 \dots 15$, and choose the overall best NMI value (cf. Section 2.2.2). Figure 7.8 displays the mean NMI value of ten iterations of the clustering results. Here FOSSCLU demonstrates its robustness against noise and its high effectiveness. All comparison methods already deteriorate after adding few noise dimensions. FOSSCLU is also reliable when increasing variance.

In addition to the synthetic data we performed experiments to evaluate the cluster quality on three real datasets, WINE ($n = 178, d = 13$) and PENDIGITS ($n = 10992, d = 16$) from the UCI Machine Learning Repository [DG19], and METABOLIC ($n = 600, d = 11$) from a PKU newborn screening [Lie+02]. Figure 7.2 shows the mean NMI of ten runs (and the corresponding standard deviation). FOSSCLU clearly outperforms all comparison methods on the WINE dataset by a large margin with an NMI of 0.87 (compared to 0.45 for LDA-K-means and EM after PCA, all others even worse. ORCLUS achieved its best results parameterized with $k = 3$ and $l = 2$ and 4C using $\epsilon = 5.6$, $MinPts = 2$ and $\lambda = 13$). All methods are outperformed with statistical significance.

The metabolic data originates from a screening program for metabolic disorders in newborns. Each instance is represented by 11 attributes representing metabolite concentrations. Out of 600 subjects 296 are from the group of healthy

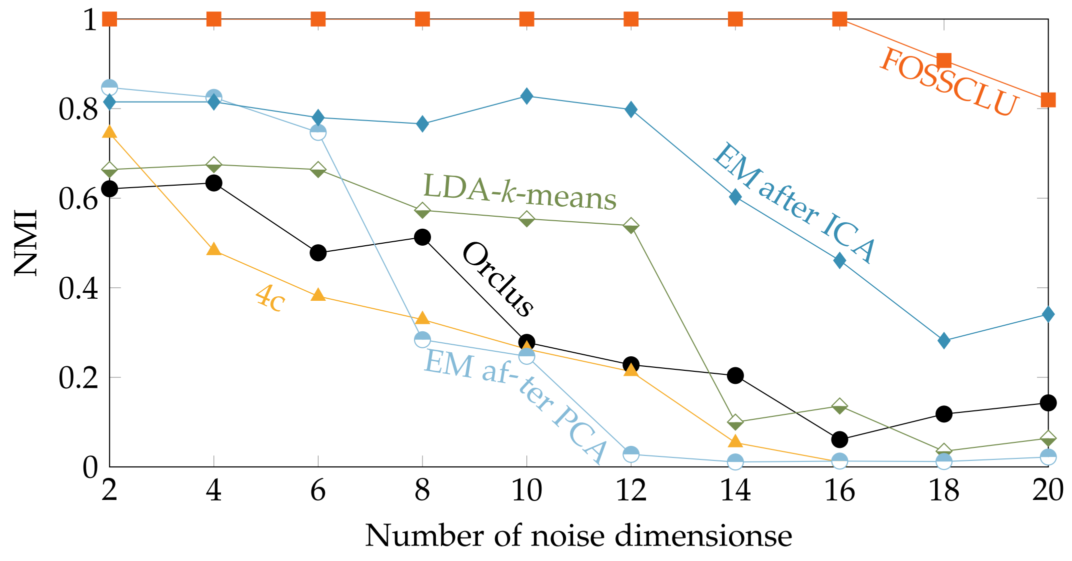


(a) Selecting the number of clusters for FOSSCLU by MDL

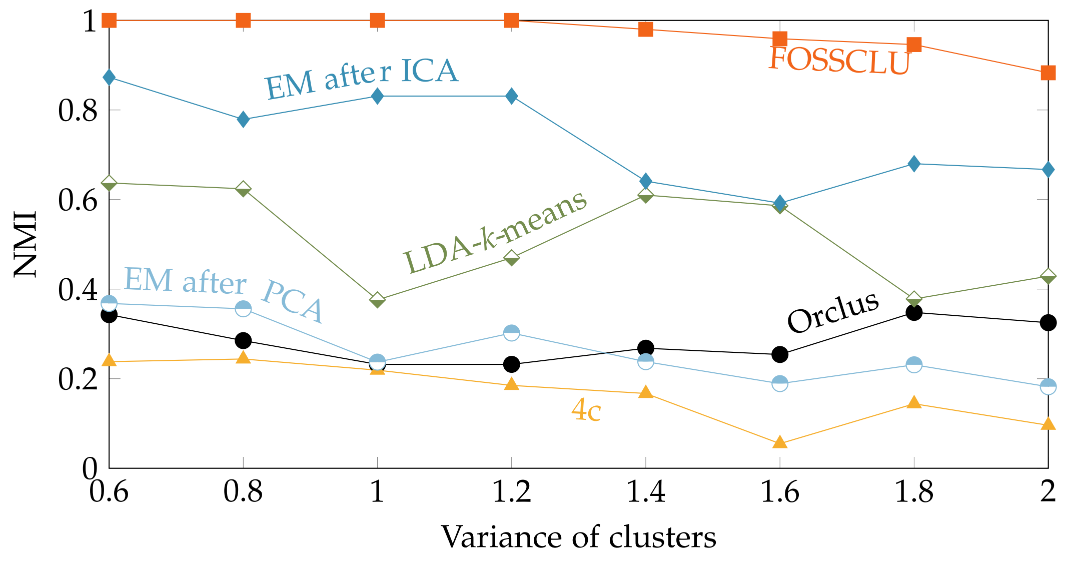


(b) Selecting the subspace dimensionality for FOSSCLU by MDL

Figure 7.7.: Selecting number of clusters and subspace dimensionality by MDL.



(a) Varying the number of noise dimensions of SYNTHETIC dataset



(b) Varying the variance of SYNTHETIC dataset

Figure 7.8.: Varying the number of noise dimensions and the variance of the clusters. We use the same SYNTHETIC dataset as in Figure 7.4 but with (a) a number of noise dimensions varied from 2 to 20 and (b) a cluster variance multiplied with a factor varying from 0.6 to 2.0

control, while the other 306 suffer from the disorder Phenylketonuria (PKU). FOSSCLU outputs two clusters of high purity (NMI = 0.88). The best parametrization for ORCLUS is $k = 2$, $l = 1$ and for 4C $\epsilon = 60$, $MinPts = 10$ and $\lambda = 11$. Our best competitor in this experiment, EM after PCA, achieved an NMI value of 0.84 but was also outperformed with statistical significance ($\alpha < 0.0002$).

In the UCI PENDIGITS dataset, FOSSCLU achieves a NMI of 0.77, followed by LDA-kMeans with a NMI of only 0.71 (FOSSCLU was significantly better: $\alpha < 0.007$).

7.4.3. Fully Automatic Parametrization

We introduced an MDL criterion in Section 7.2.4 to find the optimal parameters k and m for FOSSCLU. We first consider a synthetic dataset consisting of 5 clusters, 2 clustered dimensions and 10 noise dimensions before random rotation has been applied. The dataset is evaluated for accurate clustering. Figure 7.7 presents the cost for MDL encoding. As k increases from 1 to 10 clusters (with fixed m) and m runs through all 12 dimensions (with fixed k), MDL encoding reaches its clear minimum at $k = 5$ and $m = 2$, the correct value of the dataset. MDL performs optimal parametrization, resulting in an NMI of 1. The WINE data as introduced above also is processed by MDL analogously for k and m (Figure 7.7). The minimum of MDL coding cost is clearly found for $k = 3$ clusters. It equals the real underlying three wine cultivars. As additional benefit, MDL illustrates that 5 clusters represent the data still better than 4 clusters. Varying m with fixed $k = 3$ MDL shows optimal clustering results for $m = 7$. This choice of parameters allows the competitive NMI gained by FOSSCLU as presented in comparison to other methods.

7.5. Conclusion

We introduce FOSSCLU, a novel approach to explorative data analysis. With dimensionality reduction and clustering FOSSCLU addresses the two most important tasks in explorative data mining as two goals of equal importance. At the core of FOSSCLU is the optimal rigid transform (ORT), a new feature transformation technique particularly designed to determine the optimal subspace revealing the overall cluster structure of the data. We integrate ORT into an efficient alternating least squares algorithm to find the clusters and the subspace simultaneously. Ideas from minimum description length make parametrization in FOSSCLU optional. Our experiments demonstrate that in the resultant subspace the cluster structure is much more visible than when using standard dimensionality reduction techniques. In addition, FOSSCLU outperforms state-of-the-art approaches to generalized subspace clustering in terms of cluster quality. We can conclude that FOSSCLU combines the best of both worlds, dimensionality reduction and clustering and is an interesting complement to existing explorative data analysis approaches.

8. INTEGRATE: Integrative Parameter-free Clustering of Data with Mixed-Type Attributes

Integrative mining of heterogeneous data is one of the major challenges for data mining. We address the problem of integrative clustering of data with mixed-type attributes. Most existing solutions suffer from one or both of the following drawbacks: Either they require input parameters that are difficult to estimate, or/and they do not adequately support mixed-type attributes. Our technique INTEGRATE is a novel clustering approach that truly integrates the information provided by heterogeneous numerical and categorical attributes. Originating from information theory, the minimum description length (MDL) principle allows a unified view on numerical and categorical information and thus naturally balances the influence of both sources of information in clustering.

Moreover, supported by the MDL principle, parameter-free clustering can be performed, enhancing the usability of INTEGRATE on real-world data. Extensive experiments demonstrate the effectiveness of INTEGRATE in exploiting numerical and categorical information for clustering. As an efficient iterative algorithm, INTEGRATE is scalable to large data sets.

8.1. Introduction

Integrative data mining is among the top 10 challenging problems in data mining identified in [YW06]. Moreover, it is essential for solving many of the other top 10 challenges, including data mining in social networks and data mining for biological and environmental problems. In this chapter, we focus on *integrative clustering*. Clustering aims at finding a natural partitioning of the dataset into meaningful groups or clusters. Thus, clustering provides an overview of significant patterns in the data without requiring much previous knowledge. During the last decades, clustering has attracted much attention as reflected in a massive volume of research papers, e.g., [Böh+06; BFP08; Hua98; Mac67; PM00; YT05; ZRL96], to mention a few. We address the question of how to find a natural clustering of data with mixed-type attributes. In everyday life, vast amounts of such data are collected, for example, from credit assessments. The collected data include numerical attributes (e.g., credit amount or age) as well as categorical attributes (e.g., personal status). A cluster analysis of credit assessment data is interesting for sectors like target marketing. However, finding a natural clustering of such data is a non-trivial task. We identified two vital problems: Either much previous knowledge is required, or there is no adequate support of mixed-type attributes.

To cope with these two problems, we propose INTEGRATE, a parameter-free technique for integrative clustering of data with mixed-type attributes. The notable benefits of our approach, which, to the best of our knowledge, no other clustering method meets all of, can be summarized as follows:

1. A natural balance of numerical and categorical information in clustering supported by information theory;

2. Parameter-free clustering;

3. Making the most effective usage of numerical as well as categorical information;

4. Scalability to large datasets.

A summary of the related work has already been given in Chapter 6. The rest of this chapter is organized as follows: Section 8.2 presents a detailed derivation of *iMDL*, an information-theoretic clustering quality criterion suitable for integrative clustering. Section 8.3 presents our fast and effective iterative algorithm INTEGRATE optimizing *iMDL*. Section 8.4 documents that INTEGRATE makes the most effective usage of numerical as well as categorical information by comparing it to well-known state-of-the-art clustering algorithms on synthetic and real datasets. Section 8.5 summarizes the chapter.

8.2. Minimum Description Length for Integrative Clustering

8.2.1. Notations

In the following we consider a dataset $\mathcal{D}$ with n objects. Each object x is represented by d attributes. Attributes are denoted by capital letters and can be either numerical features or categorical variables with two or more values. For a categorical attribute A, we denote a possible value of A by a. The result of our algorithm is a disjoint partitioning of $\mathcal{D}$ into k clusters $\mathcal{C}_1, ..., \mathcal{C}_k$, so that $\mathcal{D} = \mathcal{C}_1 \dot{\cup} \cdots \dot{\cup} \mathcal{C}_k$.

8.2.2. Likelihood and Data Compression

One of the most challenging problems in clustering data with mixed-type attributes is selecting a suitable distance function or unifying clustering results obtained on the different representations of the data. Often, the weighting between the different attribute types needs to be specified by parameter settings (cf. Chapter 6). The minimum description length (MDL) principle provides a theoretical foundation for parameter-free integrative clustering avoiding this problem. Regarding clustering as a data compression problem allows us a unifying view, naturally balancing the influence of categorical and numerical attributes in clustering. Probably the most important idea of MDL, which allows integrative clustering, is relating the concepts of likelihood and data compression. Data compression can be maximized by assigning short descriptions to regular data objects which exhibit the characteristic patterns and longer descriptions to the few irregular objects or outliers.

8.2.3. Coding Categorical Data

Assume a dataset, where each object is represented by one categorical attribute A with two possible values. It can be shown that the code length to encode this data is lower bounded by the entropy of A. Thus, the coding costs CC of A are provided by:

$$CC(A) = -\sum_{a \in A} p(a) \cdot \log_2 p(a). \tag{8.1}$$

By the application of the binary logarithm, we obtain the code length in bits. If we have no additional knowledge of the data, we have to assume that the probabilities for each value are equal. Hence, we need one bit per data object. Clustering, however, provides high-level knowledge on the data which allows for a much more effective way to reduce the costs. Even if the probabilities for the different outcomes of the attributes are approximately equal for the whole dataset, often different clusters with non-uniform probabilities can be found.

For a toy example, refer to Figure 8.1. The data is represented by two numerical attributes (which we ignore for the moment) and one categorical attribute which has two possible values, *red* and *blue*. Considering all objects, the probabilities for *red* and *blue* are equal. However, it is evident that the outcomes are not uniformly distributed. Instead, we have two clusters. One preliminarily hosts the red objects, and the other the blue ones. In fact, the data has been generated such that in the left cluster, we have 88% of blue objects and 12% of red objects. For the right cluster, the ratio has been selected reciprocally. This clustering drastically reduces the entropy and hence

$$CC(A) = -(0.88 \cdot \log_2 0.88 + 0.12 \cdot \log_2 0.12) = 0.53 \text{ bits} \tag{8.2}$$

per data object, which corresponds to the entropy of A in both clusters.

8.2.4. Coding Numerical Data

To specify the probability of each data object considering an additional numerical attribute B, we assign a *probability density function* (PDF) to B. For our method, we apply a Gaussian PDF for each numerical attribute. However, let us note that

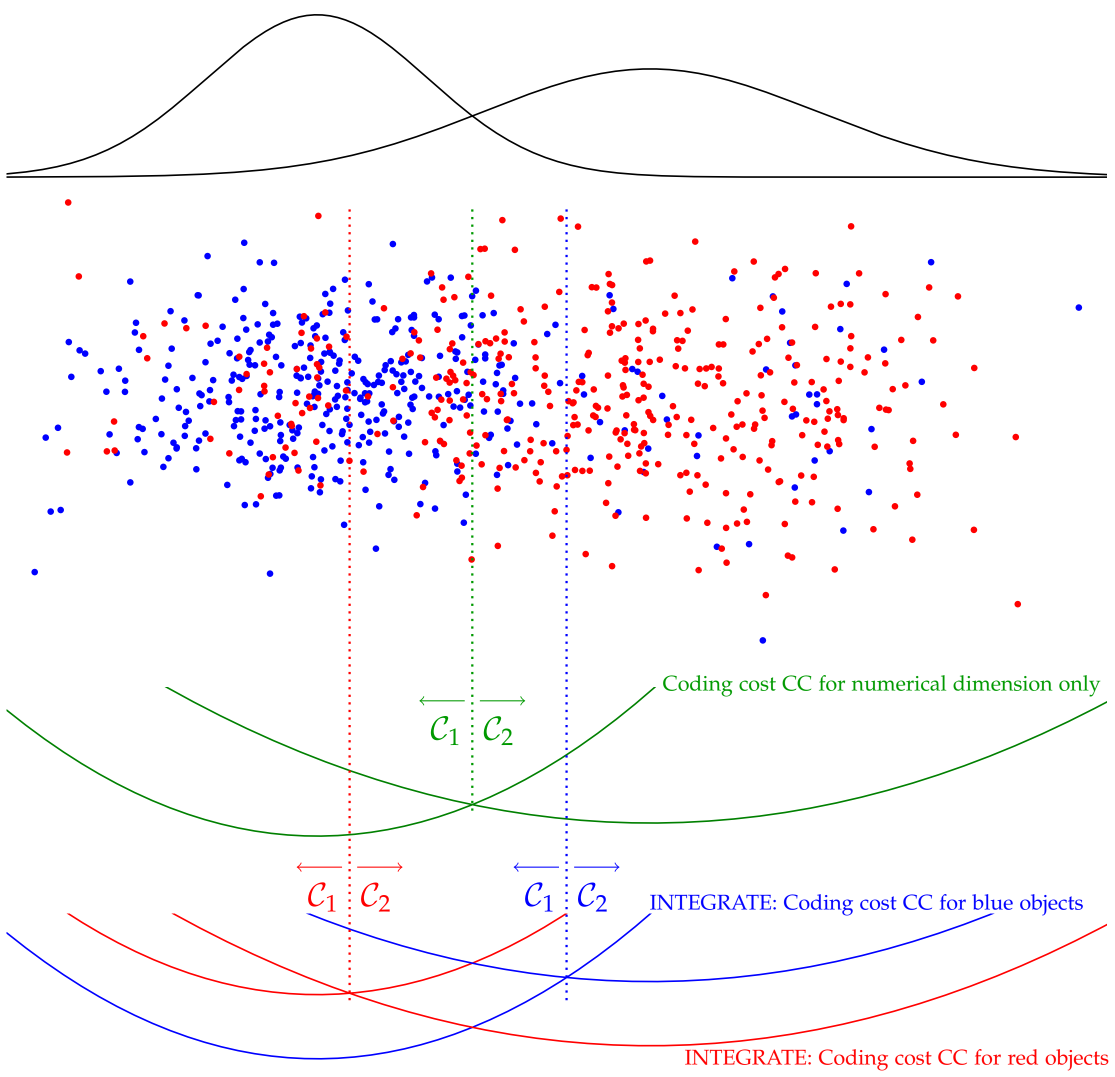


Figure 8.1.: **Toy example.** Dataset with two numerical and one categorical dimension. Numerical attributes are generated by random variables with Gaussian distributions ($\sigma = 1.0$ resp. 1.5). Categorical attribute (COLOR = {red, blue}) is generated by uniformly distributed random variables with $p(\text{red}) = 0.12$ resp. $p(\text{blue}) = 0.88$ (left cluster) and $p(\text{red}) = 0.88$ resp. $p(\text{blue}) = 0.12$ (right cluster). Cost curves assuming two clusters: Considering numerical information only (green), integrating numerical and categorical information (red, blue). Dotted lines mark resulting cluster borders.

our ideas can be straightforwardly extended to other types PDF, e.g. Laplacian or Generalized Gaussian. The PDF of a numerical attribute B is defined as

$$p(b; \mu_B, \sigma_B) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{(b-\mu_B)^2}{2 \cdot \sigma_B^2}\right) \tag{8.3}$$

with mean μ_B and standard deviation σ_B.

If the data distribution of B is Gaussian, we minimize the costs of the data compression by a coding scheme which assigns short bit strings to objects with coordinate values that are in the area of high probability and longer bit strings to objects with lower probability. This principle is also commonly referred to as Huffman coding. The coding costs CC of B are provided by:

$$CC(B) = -\int p(b) \cdot \log_2 p(b)\ \mathbf{d}b. \tag{8.4}$$

Again, if we have no prior knowledge on the data, we would have to assume that each attribute is represented by a single Gaussian with mean and standard deviation determined from all data objects. As discussed for categorical data, clustering can often drastically reduce the costs. Most importantly, relating clustering to data compression allows us a unified view on data with mixed-type attributes. Consider again the data displayed in Figure 8.1. In addition to the categorical attribute we now consider the numerical x-coordinate, denoted by X. To facilitate presentation, we ignore the y-coordinate which is processed analogously. The two green curves represent the coding costs of the two clusters considering X. For both curves, the cost minimum coincides with the mean of the Gaussian which generated the data. The cluster on the right has been generated with slightly larger variance, resulting in slightly higher coding costs. The intersection of both cost curves represents the border between the two clusters provided by X, indicated by the dotted green vertical line. In addition, for each cluster and each outcome of the categorical attribute, we have included a cost curve (displayed in the corresponding colors). Again, the intersection points mark the cluster borders provided by the categorical attribute. Consider, e.g., the dotted red vertical line. Red objects with a value in X beyond that point are assigned to the cluster on the right. Thus, in the area between the dotted red and the dotted blue vertical lines, the categorical value is the key information for clustering. Note that all borders are not fixed but optimized during the run of our algorithm.

8.2.5. A Coding Scheme for Integrative Clustering

We also need to elaborate a coding scheme describing the clustering result itself. The additional costs for encoding the clustering result can be classified into two categories: the parameter costs PC required to specify the cluster model and the ID costs IDC required to specify the cluster ID for each object, i.e., the information to which cluster the object belongs.

For the parameter costs, let us focus on the set of objects belonging to a single cluster $\mathcal{C}$. To specify the cluster model, for each categorical attribute A we need to encode the probability of each value or outcome a. For a categorical attribute with $|A|$ possible values, we need to encode $|A| - 1$ probabilities since the remaining probability is implicitly specified. For each numerical attribute B, we need to encode the parameters μ_B and σ_B of the PDF. Following a central result from the theory of MDL [Ris05], the parameter costs to model the $|\mathcal{C}|$ objects of the cluster can be approximated by $p/2 \cdot \log_2 |\mathcal{C}|$, where p denotes the number of parameters. The parameter costs logarithmically depend on the number of objects in the cluster. The considerations behind this are that the parameters do not need to be coded with very high precision for clusters with few objects. To summarize, the parameter costs for a cluster $\mathcal{C}$ are defined as

$$PC(\mathcal{C}) = \frac{1}{2} \cdot ((\sum_{A_{cat}} |A| - 1) + |B_{num}| \cdot 2)) \cdot \log_2 |\mathcal{C}|. \tag{8.5}$$

Here A_{cat} stands for all categorical attributes and B_{num} for all numerical attributes in the data. Besides the parameter costs, for each object we need to encode the information to which of the k clusters it belongs. Also for the ID costs, we apply the principle of Huffman coding which implies that we assign shorter bit-strings to the larger clusters. Thus, the ID costs of a cluster $\mathcal{C}$ are defined as

$$IDC(\mathcal{C}) = \log_2 \frac{n}{|\mathcal{C}|}. \tag{8.6}$$

Putting the parts together, we now define *iMDL*, our information-theoretic optimization goal for integrative clustering, as

$$iMDL = \sum_{\mathcal{C}} \Big(\sum_{A} |\mathcal{C}| \cdot CC(A) \Big) + PC(\mathcal{C}) + IDC(\mathcal{C}). \qquad (8.7)$$

For all clusters $\mathcal{C}$, we sum up the coding costs for all numerical and categorical attributes A. To these costs we need to add the parameter costs and the ID cost of the cluster, denoted by $PC(\mathcal{C})$ and $IDC(\mathcal{C})$, respectively. Finally, we sum up these three terms for all clusters.

8.3. The Algorithm INTEGRATE

In this section, we present our highly effective algorithm INTEGRATE for clustering mixed-type attributes which is based on our new MDL criterion *iMDL*, defined in Section 8.2. INTEGRATE is designed to find the optimal clustering of a dataset $\mathcal{D}$ where each object x comprises both numerical and categorical attributes by optimizing the overall compression rate. First, INTEGRATE builds an initial partitioning of k clusters $\mathcal{C}_1, ..., \mathcal{C}_k$. Each cluster is represented by a Gaussian PDF with mean μ_B and standard deviation σ_B in each numerical dimension B and a probability $p(a)$ for each value of each categorical attribute A. All objects are then assigned to the k clusters by minimizing the overall coding costs *iMDL*. In the next step, the parameters of each cluster are recalculated according to the assigned objects. That includes μ_B and σ_B in each numerical dimension and the probabilities $p(a)$ for each value of the categorical attributes, respectively. After this initialization the following steps are performed repeatedly until convergence: First, the costs for coding the actual cluster partition are determined. Second, the assignment of objects to clusters is performed in order to decrease the *iMDL* value. Third, the new parameters of each cluster are recalculated. INTEGRATE converges if no further changes of cluster assignments occur. Finally, we receive the optimal clustering $\mathcal{C}_1, ..., \mathcal{C}_k$ for $\mathcal{D}$ represented by k clusters according to minimum coding costs *iMDL*.

8.3.1. Initialization

The effectiveness of an algorithm often heavily depends on the quality of the initialization, as it is often the case that the algorithm can get stuck in a local optimum. Hence, we propose an initialization scheme to avoid this effect. We need to find initial cluster representatives that correspond well to the final representatives. An established method for partitioning methods is to initialize with randomly chosen objects of $\mathcal{D}$. We adopt this idea and take the numerical attributes of k randomly chosen objects as cluster representatives. During initialization, we set $\sigma = 1.0$ in each numerical dimension. The probabilities of the values for the categorical attributes are set to $\frac{1}{|A|}$, with $|A|$ being the number of possible values for A. Then, a random set of $\frac{1}{z}n$ objects is selected, where n is the number of all objects in $\mathcal{D}$ and $z = 10$ turned out to give satisfying results. We assign each object to a cluster by minimizing *iMDL*. Finally, we chose the clustering result that minimizes *iMDL* best, within m initialization runs. Typically $m = 100$ runs suffice for an effective result. As only a fraction of $\mathcal{D}$ is used for the initialization procedure, our method is not only effective but also very efficient: the time complexity of the initialization step is $O(\frac{n}{z} \cdot k \cdot m)$.

8.3.2. Automatically Selecting the Number of Clusters k

Now we propose a further improvement of the effectiveness of INTEGRATE. Using *iMDL* for mixed-type data we can avoid the parameter k. As an optimal clustering that represents the underlying data structure best has minimum coding costs, *iMDL* can also be used to detect the number of clusters. For this purpose, INTEGRATE uses *iMDL* no longer exclusively as selection criterion for finding the correct object to cluster assignment. Rather we now estimate the coding costs for each k where k is selected in a range of $1 \leq k \leq n$. For efficiency reasons INTEGRATE performs this iteration step on a $z\%$ sample of $\mathcal{D}$. The global minimum of this cost function gives the optimal k and thus the optimal number of clusters.

8.4. Experimental Evaluation

Since INTEGRATE is a hybrid approach which is combining the benefits of those clustering methods using only numeric attributes and those for categorical attributes we compare to algorithms of both categories and to algorithms that can also handle mixed-type attributes. In particular, we select

(1) the popular *k*-**means** algorithm,

(2) the widely used method *k*-**modes**,

(3) the *k*-means-based method by Ahmad and Dey denoted by *k*-**MM**, and

(4) *k*-**prototype** [Hua98] (cf. Chapter 6).

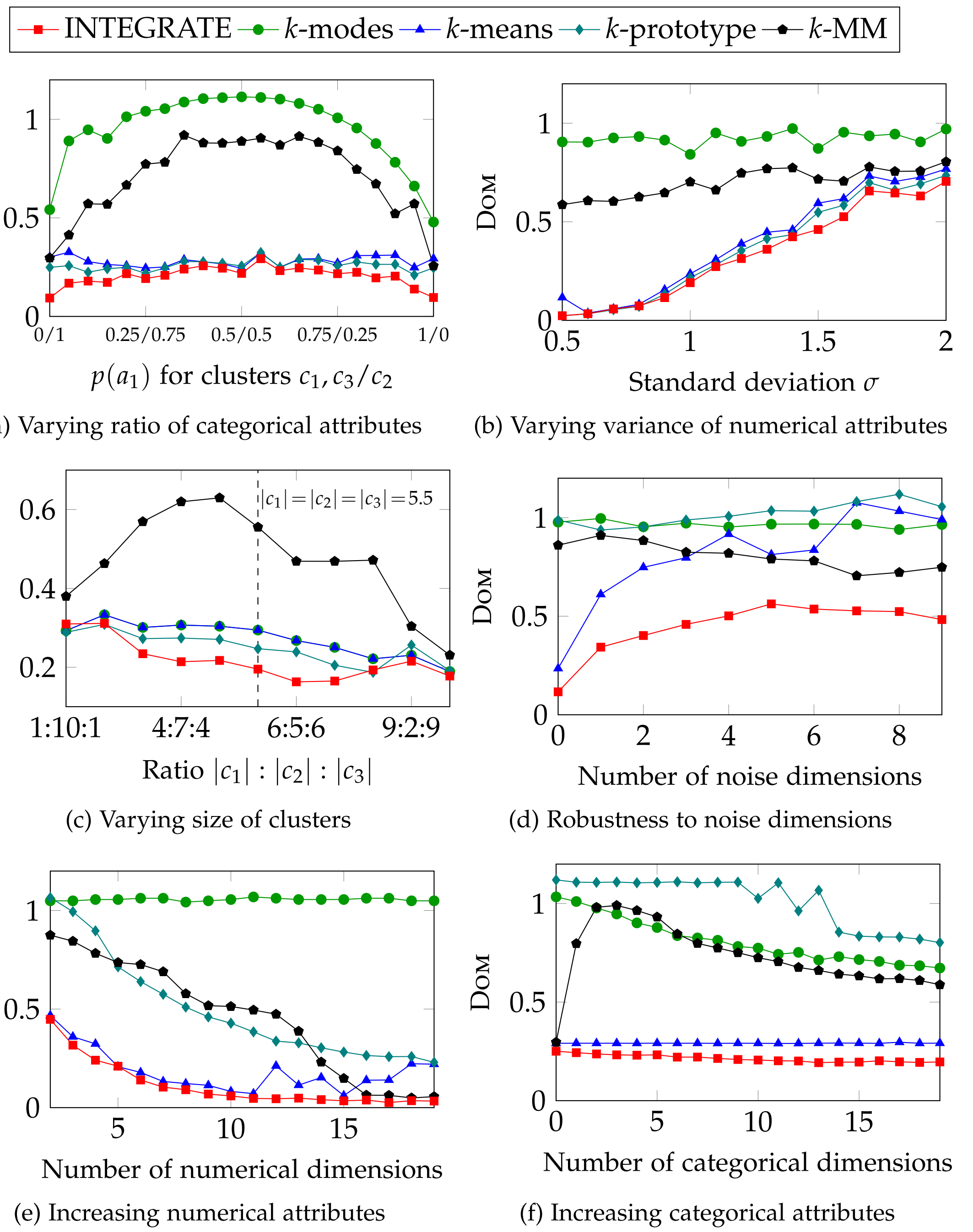
For *k*-means resp. *k*-modes only the numerical resp. categorical attributes were regarded. We evaluate the results using the information-theoretic external cluster-validity measure introduced by B. E. Dom [Dom02] (referred to as DOM) which has the advantage that it allows for clusterings with different numbers of clusters and integrates the class labels as "ground truth". The measure DOM has be introduced more detailed in Section 2.2.2. For each experiment we report the average performance of all clustering algorithms over 10 runs.

8.4.1. Synthetic Datasets

If not otherwise specified the artificial datasets include three Gaussian clusters with each object having two numerical attributes and one categorical attribute. To validate the results we added a class label to each object which was not used for clustering.

Varying the Ratio of Categorical Attribute Values (Figure 8.2a)

We create three three-dimensional clusters c_1, c_2 and c_3, each of 500 objects. Each object consists of two numerical attributes generated by a random variable with a Gaussian distribution with standard deviation $\sigma_{1,c_1} = 1.8$, $\sigma_{1,c_2} = 1.3$ and



(a) Varying ratio of categorical attributes (b) Varying variance of numerical attributes

(c) Varying size of clusters (d) Robustness to noise dimensions

(e) Increasing numerical attributes (f) Increasing categorical attributes

Figure 8.2.: **Experiments on synthetic datasets.** Measured by Dom [Dom02]: a **lower** score is better.

$\sigma_{1,c_3} = 0.8$ for the first attribute and $\sigma_{2,c_1} = 0.75$, $\sigma_{2,c_2} = 1.7$ and $\sigma_{2,c_3} = 1.0$ for the second attribute. Additionally, each object has one two-valued categorical attribute $A = \{a_1, a_2\}$ generated by a uniformly distributed random variable. In this experiment we vary the probability for each event of the random variable and, thus, the ratio of the categorical attribute values in each cluster. We start with $p(a_1) = 0$ for clusters c_1 and c_2, and with $p(a_1) = 1$ for cluster c_3, labeled as "0/1" in Figure 8.2a. Note that $p(a_2) = 1 - p(a_1)$. We increase resp. decrease $p(a_1)$ in steps of 0.05 until $p(a_1) = 1$ for clusters c_1 and c_2, and $p(a_1) = 0$ for cluster c_3.

Without the need for difficult parameter setting INTEGRATE performs best in all cases. Even in the case of equally (0.5/0.5) distributed values, where the categorical attribute gives no information for separating the objects, none of the comparison methods gives better results than INTEGRATE. As k-means does not take the categorical attributes into account the performance is relatively constant.

Varying the Variance of Clusters (Figure 8.2b)

This experiment aims at comparing the performance of INTEGRATE and its comparison methods on datasets with increasing values for variance in the numerical dimensions. We create three clusters of the same size as in the previous experiment with again two numerical and one two-valued categorical attribute. This time we fix $p(a_1) = 0.12$ for clusters c_1 and c_3, and $p(a_1) = 0.88$ for cluster c_2. The standard deviation of the underlying Gaussian distributions σ_1 and σ_2 are equally varied in every step of the experiment ($\sigma_1 = \sigma_2 = 0.5 \ldots 2.0$).

INTEGRATE clearly outperforms all competitors. Only with very small values for the standard deviation and, thus, a small degree of overlap of the three clusters, k-means and k-modes are partly able to give results of similar quality. However, at a variance of 2.0 where the numerical attributes carry nearly no cluster information our proposed method INTEGRATE shows the best cluster quality as in this case the categorical attributes are used to separate the clusters. On the contrary, k-modes performs worst as it can only use the categorical attribute as single source for clustering.

Varying the Size of the Clusters (Figure 8.2c)

We test the performance of INTEGRATE and its comparison methods on datasets with an unbalanced number of points in each cluster. As before we create three clusters of objects with two numerical and one categorical attribute. For the Gaussian distributions we set $(\sigma_{1,c_1}, \sigma_{2,c_1}) = (1.8, 0.75)$, $(\sigma_{1,c_2}, \sigma_{2,c_2}) = (1.3, 1.7)$ and $(\sigma_{1,c_3}, \sigma_{2,c_3}) = (0.8, 1)$. For the two-valued categorical attribute we set $p(a_1) = 0.12$ for cluster c_1, $p(a_1) = 0.88$ for cluster c_2 and $p(a_1) = 0.1$ for cluster c_3. We start the experiment with cluster sizes $|c_1| = 100$, $|c_2| = 1000$ and $|c_3| = 100$, which corresponds to a ratio $10 : 1 : 10$. We vary the objects per cluster by increasing resp. decreasing the number of objects in steps of 100.

INTEGRATE separates the three clusters best in almost all cases, even with highly unbalanced cluster sizes. Only in the setting with two very small clusters and one big cluster (1:10:1) k-modes gives a slightly better cluster validity. Interestingly, k-MM performs worse for balanced clusters than for unbalanced cluster.

Robustness to Noise Dimensions (Figure 8.2d)

To investigate the robustness to noise, we create three clusters of 500 objects each. The objects feature five numerical and three two-valued categorical attributes. For the underlying Gaussian distributions we set $\sigma_{i,c_1} = 1.8$, $\sigma_{i,c_2} = 1.3$ and $\sigma_{i,c_3} = 0.8$ for $i = \{1, \ldots, 5\}$. For each two-valued categorical attribute and each cluster we set $p(a_1) = 0.6$. Our experiment starts with zero noise dimensions. In each step we add an attribute to each object with values generated from a random variable with the same Gaussian distribution with $\sigma = 5$. We add up to nine noise dimensions.

INTEGRATE very clearly outperforms all comparison methods when adding the non-clustered noise dimensions to the data. The clustering quality of INTEGRATE remains stable when adding five up to nine noise dimensions. k-means shows a high increase in the Dom values which refers to decreasing cluster validity.

Increasing the Number of Numerical Dimensions (Figure 8.2e)

In this experimental setting, we increase the number of numerical attributes for each object while keeping the number of categorical attributes fixed. As before, we create three clusters of 500 objects each. Each object features only one two-valued categorical attribute with $p(a_1) = 0.3$ for clusters c_1 and c_2, and with $p(a_1) = 0.7$ for cluster c_3. The experiment starts with one numerical attribute for each object and adds one numerical dimension in each step up to 19 dimensions. For the generation of the numerical attributes for each cluster we set $\sigma_{i,c_1} = 1.8$, $\sigma_{i,c_2} = 1.3$ and $\sigma_{i,c_3} = 0.8$ for $i = \{1, \dots, 19\}$.

INTEGRATE clearly performs best in all cases. As expected, all methods demonstrate an increase in cluster quality when adding numerical dimensions except for k-modes. Since k-modes does not consider numerical attributes, its performance is constant.

Increasing the Number of Categorical Dimensions (Figure 8.2f)

Instead of the number of numerical attributes, we now increase the number of categorical attributes for each object. Each clusters consists of 500 objects with a fixed number of two numerical attributes. They are generated with the settings $(\sigma_{1,c_1}, \sigma_{2,c_1}) = (1.8, 0.75)$, $(\sigma_{1,c_2}, \sigma_{2,c_2}) = (1.3, 1.7)$ and $(\sigma_{1,c_3}, \sigma_{2,c_3}) = (0.8, 1.0)$. Our experiment starts with zero categorical dimensions and adds one categorical dimension in each step, up to 19 in the final step. Each categorical attribute is two-valued with $p(a_1) = 0.6$ for cluster c_1 and $p(a_1) = 0.2$ both for clusters c_2 and c_3.

INTEGRATE clearly outperforms all other methods, even k-modes, a state-of-the-art method for clustering categorical data. While k-MM shows a heavy decrease in clustering quality when adding one and two categorical attributes, our method performs relatively constant with even slightly increasing quality. Since k-means only takes the numerical attributes into account, its performance remains constant.

Table 8.1.: Comparison of clustering quality measured by Dom on real-world datasets. The table lists the mean and standard deviation of Dom for ten runs. Best performance printed in **bold**.

		INTEGRATE	k-means	k-modes	k-MM	k-prototype
HEART DISEASE	μ	**1.23**	1.33	1.26	1.24	1.33
	σ	0.02	0.01	0.03	0.02	0.00
CREDIT APPROVAL	μ	**0.61**	0.66	0.70	0.63	0.66
	σ	0.03	0.00	0.00	0.09	0.00

8.4.2. Real-world Datasets

This section shows the practical application of INTEGRATE on real-world data, publicly available at the UCI repository [DG19]. We chose two different datasets with mixed numerical and categorical attributes. An additional class attribute allows for an evaluation of the results. Table 8.1 reports the mean and standard deviation of the clustering quality measured with Dom for all methods for 10 runs. For all compared methods, we set k to the number of classes.

HEART DISEASE

The HEART DISEASE dataset comprises 303 instances with six numerical and eight categorical attributes each labeled by an integer value between 0 and 4 that refers to the presence of heart disease. Without any prior knowledge on the dataset, we obtained the best clustering quality of 1.23 with INTEGRATE. k-MM performed slightly worse. However, the runtime of INTEGRATE is 0.1 seconds compared to k-MM which took 2.8 seconds to return the result.

CREDIT APPROVAL

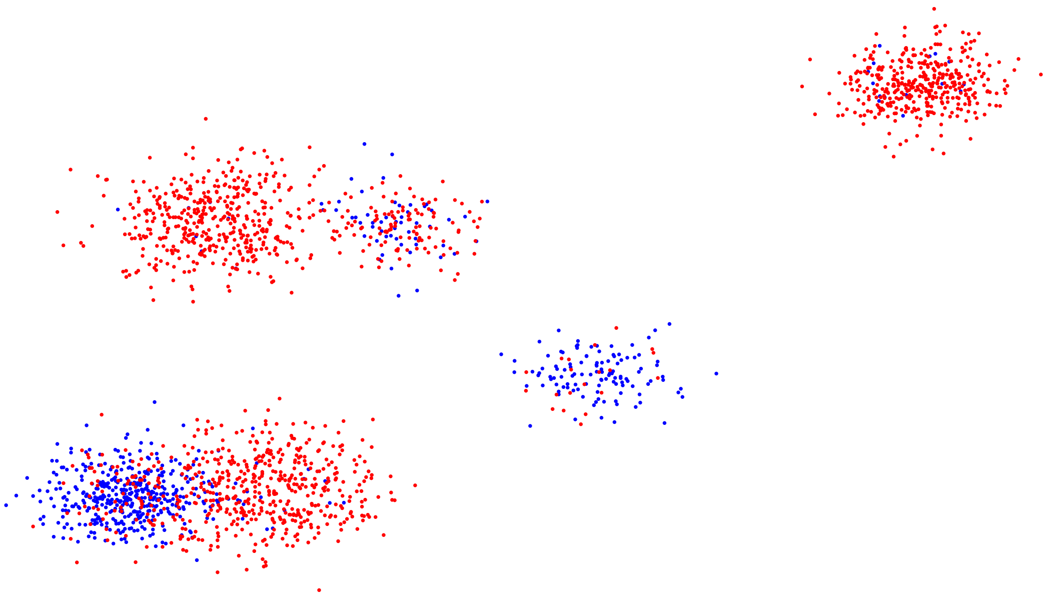
The CREDIT APPROVAL dataset contains results of credit card applications. It has 690 instances, each being described by six numerical and nine categorical attributes and classified to the two classes 'yes' or 'no'. With a mean Dom value of 0.61 INTEGRATE separated the objects best into two clusters in only 0.1 seconds without any need for setting input parameters.

8.4.3. Finding the Optimal Number of Clusters

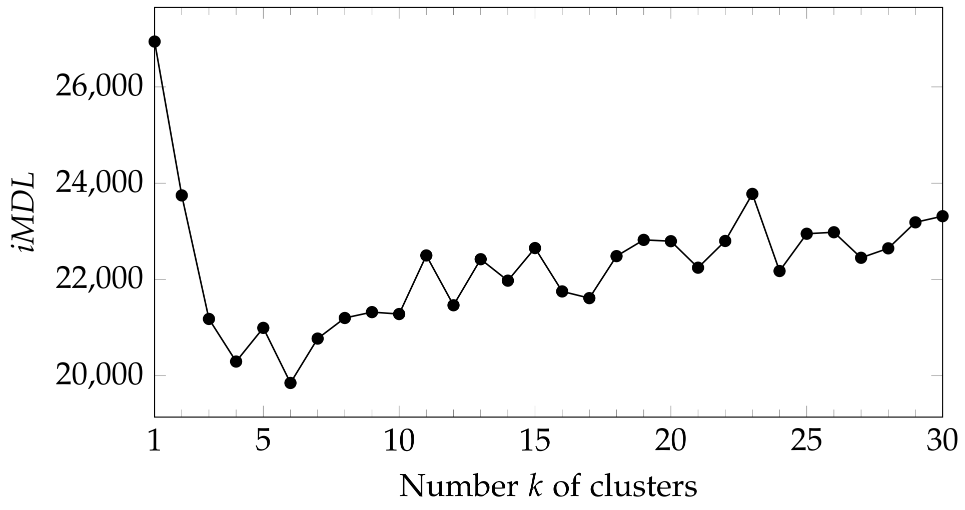
On the basis of the dataset illustrated in Figure 8.3a we highlight the benefit of INTEGRATE for finding the correct number k of clusters that are present in the dataset. The dataset comprises six Gaussian clusters with each object having two numerical attributes and one categorical attribute with two different values that are marked in "red" and "blue", respectively. Figure 8.3b shows the *iMDL* of the data model for different values of k. The cost function has its global minimum, which refers to the optimal number of clusters, at $k = 6$. In the range of $1 \leq k \leq 4$ the plotted function shows an intense decrease in the coding costs and for $k > 6$ a slight increase of the coding costs as in these cases the data does not optimally fit into the data model and, thus, leads to high coding costs. Note, that there is a local minimum at $k = 4$ which would also refer to a visually meaningful number of clusters.

8.5. Conclusion

We have introduced our information-theoretic clustering method INTEGRATE. We have given a solution to avoid complicated parameter settings guided by the information-theoretic idea of data compression. With our experiments, we have also shown that INTEGRATE uses numerical and categorical information most effectively. Finally, INTEGRATE shows high efficiency and is therefore scalable to large datasets.



(a) Synthetic dataset with 2 numerical and 1 categorical ({red, blue}) dimension



(b) *iMDL* for $1 \leq k \leq 30$

Figure 8.3.: Coding costs (*iMDL*) for different settings of k for a synthetic dataset that consists of 6 clusters.

Part IV.

Conclusion

9. Summary and Outlook

In the two main parts of this thesis, Parts II and III, we have presented our contributions to face the challenges that arise from the complexity of data and require new solutions. In this last part, we summarize our methods and conclude how they tackle which challenge. We give an outlook on future improvements to our proposed methods.

9.1. Tackling the Challenges of Mining Complex Data

The complexity of data leads to new challenges for data mining. The goal of this thesis is to contribute to tackling these challenges. Section 1.2 has formulated five key challenges derived from data complexity [HKP13; HG08; YW06]. Our four proposed algorithms contribute to facing those challenges. We summarize those challenges:

- **Challenge 1: Graphs and Network Mining.** The benefit of graphs over classic numerical or categorical vector data allows to model more complex data with various relations between data entities. Again, those relations can have attributes like weights. This increase in complexity requires new effective and efficient algorithms for pattern mining tasks with goals similar to those of clustering classic vector data.

- **Challenge 2: Explainable Data Mining.** Increasing data complexity leads to increasing complexity of the models given as output of data mining algorithms. However, a black-box model as output does not allow the user to learn the structure and patterns in the underlying data. Instead, explainable data mining focuses on transparent output and algorithmic decisions to increase the knowledge gained from the KDD process.

- **Challenge 3: High-Dimensional Data Mining.** The sheer size of data alone increases complexity: clustering algorithms have to tackle the increasing amount of noise data points that come with an increasing number of dimensions. Additionally, large datasets usually impose computational costs and require efficient solutions.

- **Challenge 4: Mining Heterogeneous Data Types.** Data is not always only numerical or cannot always be mapped to numerical dimensions that allow the benefit of using a distance function to determine similar and dissimilar data points. Datasets with numerical and categorical data types require more complex, integrated solutions for pattern mining.

- **Challenge 5: Parameter-free Data Mining.** The increasing complexity of data and the resulting data mining models make it difficult and tedious to define good input parameters that many algorithms require. Parameter-free data mining tackles this challenge by automatically finding suitable parameters without requiring user input.

Our four methods for *Fast and Effective Methods for Explainable Graph Structuring and Summarization* in Part II and for *Fast and Effective Methods for Parameter-free Clustering* in Part III all contribute to subsets of **challenges 1** to **5**. In Table 9.1, we give a concise overview of which of our contributions tackles which challenge.

In the remainder of this section, we give a summary for each of our proposed methods and discuss how each algorithm contributes to the goal of this thesis, to solve the challenges 1 to 5.

9.1.1. Summary and Discussion: Fast and Effective Methods for Explainable Graph Structuring and Summarization

In Part II, we presented our contributions to graph mining. While graphs are helpful for modeling relationships between entities, they are more complex to analyze than data represented by multi-dimensional feature vectors. As

Table 9.1.: Data mining challenges (C 1 to 5) are met by our proposed algorithms of Part II (MeGS and Spectral Lens) and Part III (FOSSCLU and INTEGRATE).

	MeGS	Spectral Lens	FOSSCLU	INTEGRATE
C 1: Graph and Network Mining	✔	✔		
C 2: Explainable Data Mining	✔	✔	✔	
C 3: High-Dimensional Data Mining		✔	✔	
C 4: Mining Heterogeneous Data Types		(✔)		✔
C 5: Parameter-free Data Mining	✔	✔	✔	✔

a contribution to all of our five challenges, in Part II, we have presented the following methods:

- MeGS for partitioning meaningful subgraph structures using Minimum Description Length (tackling challenges **C1**, **C2**, and **C5**), and
- Spectral Lens for explainable diagnostics, tools and discoveries in directed, weighted graphs (tackling challenges **C1 to C5**).

MEGS: Partitioning Meaningful Subgraph Structures using Minimum Description Length

Our first contribution is facing **challenge 1** (*graph and network mining*): Many graph partitioning methods aim to disassemble a graph into small patterns and measure good partitioning only by achieving good compression. With our proposed method MeGS for partitioning meaningful subgraph structures using minimum description length, we emphasize that patterns need to be explainable to represent valuable results. A complex graph should be fully represented by a set of patterns that practitioners can easily understand. Hence, we call these patterns *meaningful*. It is equally essential for us to analyze the graph thoroughly and assign all nodes to a pattern. No unknown or unexplained nodes should remain in the graph without a partitioning to which they are assigned. Only then can a graph be fully understood. The complete coverage of a graph with patterns and the lack of overlap between substructures allow a well-understandable and expressive visualization, being vital for *explainable data mining* (**challenge 2**)

To achieve our goal, we introduce a set of primary and simple graph structures we use as patterns. Our dictionary includes the patterns clique, bipartite structure, tree, hub, and sparse structure. If required, we could easily extend the dictionary.

Our method proposes the algorithm MeGS for partitioning a graph into subgraphs. Each subgraph corresponds to a structure of our dictionary. MeGS follows a split-and-merge principle by assigning nodes to structures, splitting and merging them successively until convergence. Partitioning a graph corresponds to permuting its adjacency matrix.

Our objective function for partitioning the graph follows the minimum description principle. The better a graph can be compressed using the structures in our dictionary, the better the partitioning result. We define an MDL schema to compress all nodes and our model parameters, rendering MeGS parameter-free. As demanded to conquer **challenge 5** (*parameter-free data mining*), the user does not need to define the number of partitions to be created. Instead, MeGS automatically finds the optimal number of partitions using our MDL schema.

We evaluate our proposed method for both efficiency and effectiveness. MeGS is a fast and efficient algorithm that outperforms its competitors both in speed and effectiveness. Since graphs often represent extensive networks in real-world applications, fast methods are essential to be of actual practical use. On several real-world datasets, we demonstrate that MeGS can explain graphs by structuring the nodes into meaningful subgraphs. We show that the visualization of the permuted adjacency matrix already gives valuable insights into how graphs are structured. Besides interpretation and as a side effect, our method is also able to compress graphs better than our competitor algorithms using our dictionary of meaningful patterns.

Limits MeGS is able to find meaningful patterns only if they are present in the dictionary. Therefore, MeGS is currently unable to identify, e.g., hyperbolic graphs as Spectral Lens does. This would require an extension of the dictionary.

Moreover, the automatic parametrization of MeGS does not always find the optimal number of clusters that gives the globally minimal minimum description length but can get stuck in a local optimum.

Spectral Lens: Explainable Diagnostics, Tools and Discoveries in Directed, Weighted Graphs

Our second contribution to **challenge 1** (*graph and network mining*) introduces our method Spectral Lens (SL) for explaining graphs, especially for understanding the connectivity patterns within and between groups of nodes, including identifying normal and anomalous behavior. Supporting *explainable data mining* (**challenge 2**), Spectral Lens visualizes all groups, their connectivity patterns, and their irregular behavior. For a given direct or undirected, weighted or unweighted graph, our proposed method SL finds groups of regular connectivity, groups that share connectivity with other groups, and suspicious behavior in the graph spectrum automatically and efficiently. Being able to process all types of graphs, including weighted and directed graphs, even allowing negative edge-weights, Spectral Lens can be seen as (although not strictly speaking) contributing to **challenge 4** (*mining heterogeneous data types*).

Our unsupervised method is based on spectral analysis. It uses singular value decomposition (SVD) to decompose large-scale graphs and extracts connectivity patterns of the left and right singular vectors. SL consists of two parts:

1. With SL-Dictionary (SLD), we provide a look-up table to understand the patterns decomposed by SVD to draw conclusions about the properties of a graph.
2. Our proposed algorithm SL-Algorithm (SLA) automatically spots:
    - the top groups of nodes with similar connectivity patterns,
    - groups of shared connectivity, and
    - groups with suspicious behavior.

For analyzing the patterns that are decomposed by SVD, we introduce two novel concepts:

- A *generalized community* (GenCom) is a group of nodes with a similar connectivity pattern. E.g., a hyperbolic community (i.e., a community with a power law distribution), a clique, or a bipartite set.
- A *bridge* is a group of nodes that shares the connectivity patterns of two GenComs but does not form a GenCom itself. Bridges result from the overlap of two or more groups and connect these groups.

In a fully connected clique, all nodes share the same neighbors and thus show the same connectivity; in a bipartite graph, both bipartite sets show similar connectivity since they only possess edges to the other set. A real-world example of a bridge is interdisciplinary conferences in computer science where researchers publish who otherwise publish to separate sets of conferences.

While SLD helps to read the plots from SVD, our algorithm SLA uses independent component analysis (ICA) to remove the tilt in the plots and automatically finds the optimal number of GenComs using our proposed measure *thorniness*. Our internal objective function frees the user from giving an input parameter for the number of groups, allowing *parameter-free data mining* (**challenge 5**). SLA also detects groups with suspicious behavior by comparing the connectivity patterns of a GenCom.

In our evaluation, we show that Spectral Lens is a valuable tool to gain insights into large-scale graphs and works on various graph types. Analyzing negative and positive edge weights, it detects fraudulent shill accounts in a real-world dataset and outperforms its competitor algorithms.

SLA is highly scalable and linear on the size of the graph. Analyzing a graph with over 2 million edges takes less than 5 minutes on a personal computer, contributing to **challenge 3** (*high-dimensional data mining*) in the dimensions of nodes and edges.

Limits The limitations of Spectral Lens are similar to those of MeGS. The thorniness heuristic to find the optimal number of in-GenComs and out-GenComs might not find the global maximum, but the parametrization might be stuck in a local maximum. Also, Spectral Lens can only explain connectivity patterns present in the SL-Dictionary.

9.1.2. Summary and Discussion: Fast and Efficient Methods for Parameter-free Clustering

In Part III, we presented our contributions to clustering. In the following, we will demonstrate how they relate to the challenges of *explainable data mining* (**challenge 2**), *high-dimensional data mining* (**challenge 3**), *mining heterogeneous data types* (**challenge 4**), and *parameter-free data mining* (**challenge 5**). Our proposed methods are

- FOSSCLU for finding the optimal subspace for clustering (tackling **C2**, **C3**, and **C5**), and
- INTEGRATE for integrative parameter-free clustering of data with mixed-type attributes (tackling **C4** and **C5**).

FOSSCLU: Finding the Optimal Subspace for Clustering

We propose our algorithm FOSSCLU (Finding the Optimal Subspace for Clustering) to simultaneously perform clustering and dimensionality reduction in a joint, alternating process to find clusters residing in low-dimensional subspaces. This process allows to filter out a high amount of noise caused by the increasing dimensionality and meets **challenge 3** (*high-dimensional data mining*). The low-dimensional reduction allows the user to visualize not only the intra-cluster relations between objects of the same clusters. Moreover, the joint subspace also preserves the inter-cluster relations of objects of separate clusters, unlike clustering algorithms that determine an individual subspace for each cluster. With FOSSCLU, it is possible to visualize the complete cluster structure in a joint low-dimensional subspace, e.g., in a two-dimensional plot to interpret the results. Here, *explainable data mining* (**challenge 2**) allows the user to understand the algorithm's decisions and provides further knowledge about the underlying data.

As a foundation for our method, we define the *optimal rigid transform* (ORT) to find an arbitrarily-oriented subspace for a given cluster structure. The key idea of ORT is to find an orthogonal pair of subspaces:

- The *clustered subspace* contains all relevant information for clustering the objects.
- The *noise subspace* contains all other information that is not relevant for clustering.

ORT applies a rigid transform, a rotation that maximizes the cluster quality in the clustered subspace. Simultaneously, the noise subspace becomes as unimodal as possible. FOSSCLU uses the expectation-maximization (EM) clustering algorithm to cluster the objects in the clustered subspace. Maximizing the cluster quality refers to an optimal EM clustering.

Our evaluation shows that FOSSCLU is comparable to state-of-the-art subspace clustering methods and can outperform them. Unlike its competitor algorithms, FOSSCLU is able to provide a joint subspace that allows interpreting the clustering results.

The user of FOSSCLU can give the number of clusters and the number of subspaces as input if a specific result is required. FOSSCLU also runs *parameter-free* due to our minimum description length (MDL)-based encoding schema. It allows FOSSCLU to find the optimal number of clusters and subspaces during the simultaneous steps of clustering and subspace reduction. Therefore, FOSSCLU requires no input parameters, tackling the challenge of *parameter-free data mining* (**challenge 5**).

FOSSCLU combines the best of both worlds: dimensionality reduction and clustering complement each other to achieve superior results in explorative data analysis.

Limits The clustering model of FOSSCLU assumes Gaussian cluster distribution and non-Gaussian noise, which might restrict its use for some applications. Furthermore, being a heuristic, FOSSCLU's automatic parametrization does not always find the globally best number of clusters and dimensions in a reasonable running time.

INTEGRATE: Integrative Parameter-free Clustering of Data with Mixed-Type Attributes

Our proposed algorithm INTEGRATE introduces a method for clustering *data with heterogeneous data types* (challenge 4). While many algorithms exist that allow for clustering datasets with only numerical dimensions or datasets with only categorical dimensions, INTEGRATE is an integrated clustering algorithm that equally considers numerical and categorical dimensions.

We introduce our minimum description length (MDL)-based coding schema *iMDL* to balance encoding costs for numerical and categorical probability distributions. Our coding schema combines the compression of model and data and guides our algorithm INTEGRATE to an optimal clustering for mixed-type data.

Our MDL-based objective function gives another benefit: apart from finding the best assignments of objects to clusters for a given number of clusters, it also serves to determine the optimal number of clusters. Thus, the user does not need to give this number as an input parameter, but it is calculated by INTEGRATE, tackling challenge 5 (parameter-free data mining).

Our synthetic experiments show that INTEGRATE is robust to noise and to unbalanced clusters and works well for clusters with a different standard deviation in the numerical dimensions. We show that INTEGRATE clearly outperforms its competitor algorithms. With our experiments on real-world datasets, we illustrate that INTEGRATE is able to show its advantage here as well by using both numerical and categorical dimensions for clustering.

Limits Like all of our methods, the automatic parametrization for INTEGRATE does not always find the globally best number of clusters to give the minimal minimum description length. Besides, INTEGRATE is not robust to noise and is restricted to Gaussian cluster models.

9.1.3. Discussion of our Contributions to Tackling the Challenges

While the previous section has discussed how each of our proposed methods helps tackle the five challenges as this thesis's contribution to fast and efficient pattern mining in complex data, we revisit each challenge in the following to decide whether the goal of this thesis has been reached.

Challenge 1: Graph and Network Mining

Our methods MeGS and Spectral Lens in Part II represent our contributions to this challenge, and they give the practitioner tools for understanding and visualizing even large and complex graphs and their patterns. MeGS focuses on partitioning all nodes in a graph into structures so that no unknown areas remain. It reveals structural information that preceding methods could not extract.

Spectral Lens improves the structural pattern detection and extends it by deriving connectivity patterns in graphs and by defining groups based on these. Spectral Lens enables the user to process not only graphs with millions of edges but also all variants of graphs: undirected and directed as well as unweighted and weighted graphs.

With both algorithmic methods, we clearly meet the goal of contributing to challenge 1.

Challenge 2: Explainable Data Mining

Creating transparent models that not only give hard black-box results but also increase the practitioner's knowledge of the inner structure of the data is essential when mining complex data and a crucial part of this thesis. We strongly believe that models must be explainable to convince the users and possibly alter their sight of the data. We contribute to this with three methods: MeGS, Spectral Lens, and FOSSCLU. All these methods visualize their results to let the user quickly grasp the findings. This holds for a sorted and structured adjacency matrix (MeGS), for the visualization of the edge-connectivity between node groups (Spectral Lens), as well as for a 2-dimensional cluster space by dropping noise dimensions (FOSSCLU). Thus, our contributions meet challenge 2.

Challenge 3: High-Dimensional Data Mining

Many well-designed algorithms in data mining might give excellent results. However, it is crucial that these methods not only work on small to medium datasets but that they can deal with data that are complex because of sheer size. This can be either due to a high number of noise dimensions in clustering for which we provide a solution with FOSSCLU or to the vast amount of edges in a network for which we propose Spectral Lens. With these contributions, we also tackle challenge 3.

Challenge 4: Mining Heterogeneous Data Types

Complexity in data also comes in the variety of data, leading to heterogeneous data types. Traditionally, many clustering algorithms offer a sophisticated model but only deal with numerical dimensions. To answer challenge 4, we propose an integrated solution for clustering data with both numerical and categorical dimensions with INTEGRATE. Additionally, for the various types of networks, we offer structural insights on undirected and directed, as well as unweighted and weighted graphs, supporting even negative edge-weights. Thus, our response to complexity in data types tackles challenge 4.

Challenge 5: Parameter-free Data Mining

Parameter-free data mining is a guiding thread throughout this thesis. We regard it as vital to achieving a very good solution without requiring the user to set parameters at the first run of any of our methods. Setting parameters requires a deeper understanding of methods and can lead to non-optimal results otherwise. However, automatic parametrization does not always find the globally optimal solution but might be stuck in a local optimum. Therefore, all our solutions allow the user to define the parameters if needed. Our automatically detected parameters are the number of groups or subgraph structures for MeGS and Spectral Lens, the number of clusters for FOSSCLU and INTEGRATE, and the number of dimensions for the clustered space in FOSSCLU. Setting the latter gives the ability to obtain an explainable 2-dimensional visualization. With these contributions, we also meet our challenge 5.

9.2. Outlook

Our contributions to the challenges in data mining that we focus on in this thesis provide the practitioner with valuable and ready-for-use methods to deal with those challenges. For all our solutions, we provide concise algorithms, sample implementations, and the data used for our evaluations (cf. Section 2.4), allowing users to benefit from this thesis immediately and mitigate the impact of the challenges to the various data mining tasks. Furthermore, this thesis contributes to ongoing challenges in data mining and can only represent a small step for science. Nevertheless, our results provide many connections for future research or have already inspired new work (cf. [Mau+17]).

Future development will include new challenges that arise from those tackled in this thesis, and with increasing complexity, existing challenges will become more difficult. While our methods are able to handle large and complex graphs as our challenge *graphs and network mining* demands even more extensive or more complicated graphs, e.g., with high-dimensional edge- or node labels as attributes, might require advanced solutions. Subsequently, further challenges in graph mining arise from, e.g., noisy data or petabyte-scale data sizes.

Our algorithm MeGS could be extended to structure more complex graphs by adding additional primitives to its dictionary together with a new MDL-encoding schema and, if necessary, an algorithmic node sorting. Beyond that, MeGS could even be improved to suggest meaningful primitives to the user. This would enhance the usefulness of MeGS for a broader range of patterns and applications.

Similarly, our second graph-mining algorithm, Spectral Lens, works on a dictionary of patterns to interpret the connectivity between nodes in a graph. Currently, our algorithm SLA can detect groups of similar connectivity, find bridges that connect groups of similar connectivity, and find suspicious connectivity patterns. However, with a further comprehensive dictionary, our algorithm could be improved to extract additional connectivity patterns for even better explainable diagnostics.

Our challenge *explainable data mining* is already becoming more critical with the increasing attention to artificial neural networks or other black-box-classifiers. With the right amount of training data, these methods are able to produce impressive results, but unlike our proposed methods, the user is often unable to reproduce the steps that the algorithm has gone through to produce a result or to understand the underlying model. Allowing model complexity and transparency at the same time will continue to be challenging for future data mining applications.

There has been much research to tackle the challenge of *high-dimensional data mining* in the past, and with FOSSCLU, we contribute another notion for a more intelligent clustering. Naturally, FOSSCLU is not the "holy grail" either and has limitations like the assumption of Gaussian clusters. Most high-dimensional clustering algorithms handle specific cases or answer specific questions. Like in many areas of data mining, a unified approach and fewer ad-hoc solutions could be a goal of future research (cf. [YW06]). For FOSSCLU, a step toward this could be the waiver of the assumption of Gaussianity.

Of the proposed clustering algorithms in this thesis, INTEGRATE tackles the challenge of *mining heterogeneous data types* but cannot exclude noise dimensions as FOSSCLU does. A logical continuation would be the combination of high-dimensional data mining with mining heterogeneous data types.

Finally, the challenge of *parameter-free data mining* is an ongoing task when developing new algorithmic solutions. While all of our methods have the ability to find a suitable parametrization without user input, they do not always find the globally optimal solution, but the optimization might be stuck at a local optimum. A future challenge is to avoid returning only a locally optimized parametrization but to give the best result possible, or at least quality guarantees on the result. However, this would still have to be computed in a reasonable time.

Obviously, all our challenges are open-end-questions and continue to be ongoing research tasks. However, we firmly believe that this thesis brings us a step closer to an answer.

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

München, den 20. Oktober 2022

Sebastian Göbl

Bibliography

[Ach+] E. Achtert, C. Böhm, J. David, P. Kröger, and A. Zimek. “Robust Clustering in Arbitrarily Oriented Subspaces”. In: *Proceedings of the 2008 SIAM International Conference on Data Mining*. SDM ’08, pp. 763–774. DOI: 10.1137/1.9781611972788.69.

[Ach+06] E. Achtert, C. Böhm, P. Kröger, and A. Zimek. “Mining Hierarchies of Correlation Clusters”. In: *18th International Conference on Scientific and Statistical Database Management*. SSDBM’06. 2006, pp. 119–128. DOI: 10.1109/SSDBM.2006.35.

[Ach+07] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. “On Exploring Complex Relationships of Correlation Clusters”. In: *19th International Conference on Scientific and Statistical Database Management*. SSDBM ’07. 2007. DOI: 10.1109/SSDBM.2007.21.

[AD07] A. Ahmad and L. Dey. “A k-mean Clustering Algorithm for Mixed Numeric and Categorical Data”. In: *Data & Knowledge Engineering* 63.2 (2007), pp. 503–527. DOI: 10.1016/j.datak.2007.03.016.

[AF09] L. Akoglu and C. Faloutsos. “RTG: A Recursive Realistic Graph Generator Using Random Typing.” In: *PKDD ’14*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 13–28. DOI: 10.1007/978-3-642-04180-8_13.

[Agg+99] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. “Fast Algorithms for Projected Clustering”. In: *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’99. 1999, pp. 61–72. DOI: 10.1145/304182.304188.

[Agg15] C. Aggarwal. *Data Mining*. Springer, 2015. DOI: 10.1007/978-3-319-14142-8.

[Agr+98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications". In: *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*. SIGMOD '98. Seattle, Washington, USA: ACM, 1998, pp. 94–105. DOI: `10.1145/276304.276314`.

[AMF10] L. Akoglu, M. McGlohon, and C. Faloutsos. "Oddball: Spotting Anomalies in Weighted Graphs." In: *PAKDD*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 410–421. DOI: `10.1007/978-3-642-13672-6_40`.

[AR13] C. C. Aggarwal and C. K. Reddy, eds. *Data Clustering: Algorithms and Applications*. CRC Press, 2013.

[Ara+14] M. Araujo, S. Günnemann, G. Mateos, and C. Faloutsos. "Beyond Blocks: Hyperbolic Community Detection." In: *ECML PKDD '14: Machine Learning and Knowledge Discovery in Databases*. 2014, pp. 50–65. DOI: `10.1007/978-3-662-44848-9_4`.

[Ass+07a] I. Assent, R. Krieger, E. Müller, and T. Seidl. "DUSC: Dimensionality Unbiased Subspace Clustering". In: *Seventh IEEE International Conference on Data Mining*. ICDM'07. IEEE, 2007, pp. 409–414. DOI: `10.1109/ICDM.2007.49`.

[Ass+07b] I. Assent, R. Krieger, E. Müller, and T. Seidl. "VISA: Visual Subspace Clustering Analysis". In: *ACM SIGKDD Explorations Newsletter* 9.2 (2007), pp. 5–12. DOI: `10.1145/1345448.1345451`.

[AY00] C. C. Aggarwal and P. S. Yu. "Finding Generalized Projected Clusters in High Dimensional Spaces". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: ACM, 2000, pp. 70–81. DOI: `10.1145/342009.335383`.

[Beu+13] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. "CopyCatch: Stopping Group Attacks by Spotting Lockstep Behavior in Social Networks". In: *WWW '13: Proceedings of the 22nd International Conference on World Wide Web*. Carnegie Mellon University. New York, New York, USA: International World Wide Web Conferences

Steering Committee, 2013, pp. 119–130. DOI: 10.1145/2488388.2488400.

[BFP08] C. Böhm, C. Faloutsos, and C. Plant. “Outlier-robust Clustering Using Independent Components”. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’08. Vancouver, Canada: ACM, 2008, pp. 185–198. DOI: 10.1145/1376616.1376638.

[Böh+04a] C. Böhm, K. Railing, H.-P. Kriegel, and P. Kroger. “Density connected clustering with local subspace preferences”. In: *Fourth IEEE International Conference on Data Mining (ICDM’04)*. 2004, pp. 27–34. DOI: 10.1109/ICDM.2004.10087.

[Böh+04b] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. “Computing Clusters of Correlation Connected Objects”. In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’04. Paris, France: ACM, 2004, pp. 455–466. DOI: 10.1145/1007568.1007620.

[Böh+06] C. Böhm, C. Faloutsos, J.-Y. Pan, and C. Plant. “Robust Information-theoretic Clustering”. In: *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’06. Philadelphia, PA, USA: ACM, 2006, pp. 65–75. DOI: 10.1145/1150402.1150414.

[Böh+10] C. Böhm, A. Oswald, C. Plant, M. Plavinski, and B. Wackersreuther. “Integrative Parameter-Free Clustering of Data with Mixed Type Attributes”. In: *PAKDD ’10: Advances in Knowledge Discovery and Data Mining*. 2010, pp. 38–47. DOI: 10.1007/978-3-642-13657-3_7.

[Bor+20] K. Borgwardt, E. Ghisu, F. Llinares-López, L. OBray, and B. Rieck. “Graph Kernels: State-of-the-Art and Future Challenges”. In: *Foundations and Trends in Machine Learning* 13.5-6 (2020), pp. 531–712. DOI: 10.1561/2200000076.

[Bro08] R. K. Brouwer. “Clustering Feature Vectors with Mixed Numerical and Categorical Attributes”. In: *International Journal of Computational Intelligence Systems* 1.4 (2008), pp. 285–298. DOI: 10.1080/18756891.2008.9727625.

[BV04] P. Boldi and S. Vigna. “The Webgraph Framework I: Compression Techniques”. In: *Proceedings of the 13th International Conference on World Wide Web*. New York, NY: ACM, 2004, pp. 595–602. DOI: 10.1145/988672.988752.

[Cha+04] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. “Fully Automatic Cross-associations”. In: *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 79–88. DOI: 10.1145/1014052.1014064.

[Chi+09] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. “On Compressing Social Networks”. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY: ACM, 2009, pp. 219–228. DOI: 10.1145/1557019.1557049.

[DG19] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2019. URL: http://archive.ics.uci.edu/ml (visited on 07/31/2019).

[DGK07] I. S. Dhillon, Y. Guan, and B. Kulis. “Weighted Graph Cuts without Eigenvectors A Multilevel Approach”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.11 (2007), pp. 1944–1957. DOI: 10.1109/TPAMI.2007.1115.

[Die17] R. Diestel. *Graph Theory*. 5th ed. Berlin, Heidelberg: Springer, 2017. DOI: 10.1007/978-3-662-53622-3.

[DL07] C. Ding and T. Li. “Adaptive Dimension Reduction Using Discriminant Analysis and K-means Clustering”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML '07. Corvalis, Oregon, USA: ACM, 2007, pp. 521–528. DOI: 10.1145/1273496.1273562.

[DMM03] I. S. Dhillon, S. Mallela, and D. S. Modha. "Information-theoretic co-clustering". In: *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. IBM Almaden Research Center. New York, New York, USA: ACM, Aug. 2003, pp. 89–98. DOI: 10.1145/956750.956764.

[Dom01] B. E. Dom. *An Information-Theoretic External Cluster-Validity Measure*. Tech. rep. Research Report RJ 10219, IBM, 2001, pp. 1–31.

[Dom02] B. E. Dom. "An Information-Theoretic External Cluster-Validity Measure". In: *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*. UAI'02. Alberta, Canada: Morgan Kaufmann Publishers Inc., 2002, pp. 137–145.

[Don08] S. van Dongen. "Graph Clustering Via a Discrete Uncoupling Process". In: *SIAM Journal on Matrix Analysis and Applications* 30.1 (Jan. 2008), pp. 121–141. DOI: 10.1137/040608635.

[Dri+] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. "Clustering Large Graphs via the Singular Value Decomposition". In: *Machine learning* 56.1-3 (), pp. 9–33. DOI: 10.1023/B%3AMACH.0000033113.59016.96.

[DuB08] C. L. DuBois. *UCI Network Data Repository*. Tech. rep. 2008.

[Dwy09] T. Dwyer. "Scalable, Versatile and Simple Constrained Graph Layout". In: *Computer Graphics Forum* 28.3 (2009), pp. 991–998. DOI: 10.1111/j.1467-8659.2009.01449.x.

[Est+96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon, 1996, pp. 226–231.

[Fen+12] J. Feng, X. He, B. Konte, C. Böhm, and C. Plant. "Summarization-based Mining Bipartite Graphs". In: *KDD '12: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2012, pp. 1249–1257. DOI: 10.1145/2339530.2339725.

[Fen+13] J. Feng, X. He, N. Hubig, C. Bohm, and C. Plant. "Compression-Based Graph Mining Exploiting Structure Primitives". In: *IEEE 13th International Conference on Data Mining*. June 2013, pp. 181–190. DOI: `10.1109/ICDM.2013.56`.

[Fou+12] F. Fouss, K. Francoisse, L. Yen, A. Pirotte, and M. Saerens. "An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification". In: *Neural Networks* 31 (2012), pp. 53–72. DOI: `10.1016/j.neunet.2012.03.001`.

[FPS96a] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "From Data Mining to Knowledge Discovery in Databases". In: *AI Magazine* 17.3 (1996), pp. 37–54. DOI: `10.1609/aimag.v17i3.1230`.

[FPS96b] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "Knowledge Discovery and Data Mining: Towards a Unifying Framework". In: *KDD '96: Proceedings of the Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1996, pp. 82–88.

[GKF17] S. Goebl, S. Kumar, and C. Faloutsos. "Spectral Lens: Explainable Diagnostics, Tools and Discoveries in Directed, Weighted Graphs". In: *ICDM '17: Proceedings of the 2017 IEEE 17th International Conference on Data Mining*. 2017, pp. 877–882. DOI: `10.1109/icdm.2017.108`.

[Goe+14] S. Goebl, X. He, C. Plant, and C. Böhm. "Finding the Optimal Subspace for Clustering". In: *ICDM '14: Proceedings of the 2014 IEEE 14th International Conference on Data Mining*. 2014, pp. 130–139. DOI: `10.1109/icdm.2014.34`.

[Goe+16] S. Goebl, A. Tonch, C. Böhm, and C. Plant. "MeGS: Partitioning Meaningful Subgraph Structures using Minimum Description Length". In: *ICDM '16: Proceedings of the 2016 IEEE 16th International Conference on Data Mining*. 2016, pp. 889–894. DOI: `10.1109/icdm.2016.0108`.

[Goo21] Google LLC. *Google Trends*. 2021. URL: `https://trends.google.de/trends/explore?date=all&q=Data%5C%20Science,Big%5C%20Data,Data%5C%20Analytics,Machine%5C%20Learning` (visited on 04/24/2021).

[Gre10] S. Gregory. “Finding Overlapping Communities in Networks by Label Propagation”. In: *New Journal of Physics* 12.1 (Oct. 2010), p. 103018. DOI: 10.1088/1367-2630/12/10/103018.

[Grü05] P. Grünwald. “A Tutorial Introduction to the Minimum Description Length Principle”. In: *Advances in Minimum Description Length: Theory and Applications*. Ed. by P. D. Grünwald, J. I. Myung, and M. A. Pitt. 2005. ISBN: 9780262072625.

[Gün+08] H. Gündel, M. Valet, C. Sorg, D. Huber, C. Zimmer, T. Sprenger, and T. R. Tölle. “Altered Cerebral Response to Noxious Heat Stimulation in Patients with Somatoform Pain Disorder.” In: *Pain* 137.2 (2008), pp. 413–421. DOI: 10.1016/j.pain.2007.10.003.

[HC07] C.-C. Hsu and Y.-C. Chen. “Mining of Mixed Data with Application to Catalog Marketing”. In: *Expert Systems with Applications* 32.1 (2007), pp. 12–23. DOI: 10.1016/j.eswa.2005.11.017.

[HG08] J. Han and J. Gao. “Research Challenges for Data Mining in Science and Engineering”. In: *Next Generation of Data Mining*. Ed. by H. Kargupta, J. Han, P. S. Yu, R. Motwani, and V. Kumar. 2008. DOI: 10.1201/9781420085877.

[HKP13] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. 3rd. San Francisco: Morgan Kaufmann Publishers Inc., 2013. ISBN: 9780123814791. DOI: 10.1016/C2009-0-61819-5.

[HO00] A. Hyvärinen and E. Oja. “Independent Component Analysis: Algorithms and Applications”. In: *Neural Networks* 13 (2000), pp. 411–430. DOI: 10.1016/S0893-6080(00)00026-5.

[Hoo+16] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos. “FRAUDAR: Bounding Graph Fraud in the Face of Camouflage”. In: *22nd ACM SIGKDD International Conference*. New York, New York, USA: ACM Press, 2016, pp. 895–904. DOI: 10.1145/2939672.2939747.

[Hua98] Z. Huang. "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values". In: *Data Mining and Knowledge Discovery* 2.3 (1998), pp. 283–304. DOI: `10.1023/A:1009769707641`.

[HXD05] Z. He, X. Xu, and S. Deng. "Clustering Mixed Numeric and Categorical Data: A Cluster Ensemble Approach". In: *Computing Research Repository* abs/cs/0509011 (2005). DOI: `10.48550/arXiv.cs/0509011`.

[Jia+14a] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang. "CatchSync: catching synchronized behavior in large directed graphs". In: *KDD '14: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. Carnegie Mellon University. ACM, Aug. 2014, pp. 941–950. DOI: `10.1145/2623330.2623632`.

[Jia+14b] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang. "Inferring Strange Behavior from Connectivity Pattern in Social Networks". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by V. S. Tseng, T. B. Ho, Z. H. Zhou, and H. Y. Kao. Cham: Springer International Publishing, May 2014, pp. 126–138. DOI: `10.1007/978-3-319-06608-0_11`.

[JNH07] L. Jing, M. K. Ng, and J. Z. Huang. "An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data". In: *IEEE Transactions on Knowledge and Data Engineering* 19.8 (2007), pp. 1026–1041. DOI: `10.1109/TKDE.2007.1048`.

[Jol02] I. T. Jolliffe. *Principal Component Analysis*. 2nd ed. Springer Series in Statistics. New York: Springer, 2002. DOI: `10.1007/b98835`.

[KF11] U. Kang and C. Faloutsos. "Beyond 'Caveman Communities': Hubs and Spokes for Graph Compression and Mining". In: *ICDM '14: Proceedings of the 2014 IEEE 14th International Conference on Data Mining*. IEEE Computer Society, 2011, pp. 300–309. DOI: `10.1109/ICDM.2011.26`.

[KHC05] N. S. Ketkar, L. B. Holder, and D. J. Cook. "Subdue: Compression-Based Frequent Pattern Discovery in Graph Data". In: *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations (OSDM'05)*. 2005, pp. 71–76. DOI: 10.1145/1133905.1133915.

[KK04] M. Kuramochi and G. Karypis. "An Efficient Algorithm for Discovering Frequent subgraphs". In: *IEEE Transactions on Knowledge and Data Engineering* 16.9 (2004), pp. 1038–1051. DOI: 10.1109/TKDE.2004.33.

[KK99] G. Karypis and V. Kumar. "Multilevel k-way Hypergraph Partitioning". In: *DAC* (1999), pp. 343–348. DOI: 10.1145/309847.309954.

[KKK04] K. Kailing, H.-P. Kriegel, and P. Kröger. "Density-Connected Subspace Clustering for High-Dimensional Data." In: *Proceedings of the 2004 SIAM International Conference on Data Mining*. SDM'04. 2004, pp. 246–256. DOI: 10.1137/1.9781611972740.23.

[KKZ09] H.-P. Kriegel, P. Kröger, and A. Zimek. "Clustering High-dimensional Data: A Survey on Subspace Clustering, Pattern-based Clustering, and Correlation Clustering". In: *ACM Transactions on Knowledge Discovery from Data* 3.1 (2009), 1:1–1:58. DOI: 10.1145/1497577.1497578.

[Kou+14] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. "VOG: Summarizing and Understanding Large Graphs". In: *SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, Apr. 2014, pp. 91–99. DOI: 10.1137/1.9781611973440.11.

[KSS14] S. Kumar, F. Spezzano, and V. S. Subrahmanian. "Accurately detecting trolls in Slashdot Zoo via decluttering". In: (2014), pp. 188–195. DOI: 10.1109/ASONAM.2014.6921581.

[KTF11] U. Kang, C. E. Tsourakakis, and C. Faloutsos. "PEGASUS: Mining Peta-Scale Graphs". In: *Knowledge and Information Systems* 27.2 (2011), pp. 303–325.

[Kum+16] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos. “Edge Weight Prediction in Weighted Signed Networks”. In: *Data Mining (ICDM), 2016 IEEE International Conference on*. 2016, pp. 221–230. DOI: `10.1109/ICDM.2016.0033`.

[LC08] T. Li and Y. Chen. “A Weight Entropy k-Means Algorithm for Clustering Dataset with Mixed Numeric and Categorical Data”. In: *Fifth International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 1. FSKD’08. 2008, pp. 36–41. DOI: `10.1109/FSKD.2008.32`.

[LC78] S. Leung-Yan-Cheong and T. M. Cover. “Some Equivalences Between Shannon Entropy and Kolmogorov Complexity”. In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 331–338. DOI: `10.1109/TIT.1978.1055891`.

[Lie+02] B. Liebl, U. Nennstiel-Ratzel, R. von Kries, R. Fingerhut, B. Olgemöller, A. Zapf, and A. A. Roscher. “Very High Compliance in an Expanded MS-MS-Based Newborn Screening Program despite Written Parental Consent”. In: *Preventive Medicine* 34.2 (2002), pp. 127–131. DOI: `10.1006/pmed.2001.0952`.

[Liu+18] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. “Graph Summarization Methods and Applications: A Survey”. In: *ACM Computing Surveys (CSUR)* 51.3 (2018). DOI: `10.1145/3186727`.

[LS00] D. D. Lee and H. S. Seung. “Algorithms for Non-negative Matrix Factorization”. In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. NIPS’00. Denver, CO: MIT Press, 2000, pp. 535–541.

[Mac67] J. MacQueen. “Some Methods for Classification and Analysis of Multivariate Observations”. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967, pp. 281–297.

[Mat+00] T. Matsuda, T. Horiuchi, H. Motoda, and T. Washio. “Extension of Graph-Based Induction for General Graph Structured Data”. In: *Lecture Notes in Computer Science*. Ed. by T. Terano, H. Liu, and A. P. Chen. Springer Berlin Heidelberg, 2000, pp. 420–431. DOI: `10.1007/3-540-45571-X_49`.

[Mau+17] D. Mautz, W. Ye, C. Plant, and C. Böhm. “Towards an Optimal Subspace for K-Means”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13-17, 2017*. ACM, 2017, pp. 365–373. DOI: `10.1145/3097983.3097989`.

[Mie15] P. Miettinen. “Generalized Matrix Factorizations as a Unifying Framework for Pattern Set Mining: Complexity Beyond Blocks”. In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II*. Cham: Springer International Publishing, 2015, pp. 36–52. DOI: `10.1007/978-3-319-23525-7_3`.

[MV14] P. Miettinen and J. Vreeken. “MDL4BMF: Minimum Description Length for Boolean Matrix Factorization”. In: *ACM Transactions on Knowledge Discovery from Data* 8.4 (Oct. 2014), 18:1–18:31. DOI: `10.1145/2601437`.

[NRS08] S. Navlakha, R. Rastogi, and N. Shrivastava. “Graph summarization with bounded error”. In: *SIGMOD International Conference on Management of Data (SIGMOD'04)*. New York, New York, USA: ACM Request Permissions, June 2008, pp. 419–432. DOI: `10.1145/1376616.1376661`.

[Orr08] W. W. Orrison. *Atlas of Brain Function*. Thieme, 2008.

[PM00] D. Pelleg and A. W. Moore. “X-means: Extending K-means with Efficient Estimation of the Number of Clusters”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 727–734.

[Pra+10] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos. "EigenSpokes: Surprising Patterns and Scalable Community Chipping in Large Graphs. " In: *PAKDD*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 435–448. DOI: 10.1007/978-3-642-13672-6_42.

[Pre+07] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes. The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

[Ris05] J. Rissanen. *An Introduction to the MDL Principle*. Tech. rep. Helsinkin Institute for Information Technology, 2005.

[Ris07] J. Rissanen. *Information and Complexity in Statistical Modeling*. Information Science and Statistics. New York, NY: Springer Science & Business Media, 2007. DOI: 10.1007/978-0-387-68812-1.

[Ris78] J. Rissanen. "Modeling by shortest data description". In: *Automatica* 14.5 (1978), pp. 465–471. DOI: 10.1016/0005-1098(78)90005-5.

[RS06] E. Rendón and J. S. Sánchez. "Clustering Based on Compressed Data for Categorical and Mixed Attributes". In: *Proceedings of the 2006 Joint IAPR International Conference on Structural, Syntactic, and Statistical Pattern Recognition*. SSPR'06/SPR'06. Hong Kong, China: Springer-Verlag, 2006, pp. 817–825. DOI: 10.1007/11815921_90.

[SD11] S. Sarkar and A. Dong. "Community Detection in Graphs using Singular Value Decomposition". In: *Physical Review E* 83.4 (Apr. 2011), pp. 1082–16. DOI: 10.1103/PhysRevE.83.046114.

[SG76] S. Sahni and T. Gonzalez. "P-complete approximation problems". In: *Journal of the ACM* 23.3 (1976), pp. 555–565. DOI: 10.1145/321958.321975.

[Sha+14] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos. "Spotting Suspicious Link Behavior with fBox: An Adversarial Perspective". In: *ICDM '14: Proceedings of the 2014 IEEE 14th International Conference on Data Mining*. IEEE, 2014, pp. 959–964. DOI: 10.1109/ICDM.2014.36.

[Sha+16] N. Shah, A. Beutel, B. Hooi, L. Akoglu, S. Günnemann, D. Makhija, M. Kumar, and C. Faloutsos. "EdgeCentric - Anomaly Detection in Edge-Attributed Networks." In: *ICDM Workshops* (2016), pp. 327–334. DOI: 10.1109/ICDMW.2016.0053.

[Sha48] C. E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[Sim+13] K. Sim, V. Gopalkrishnan, A. Zimek, and G. Cong. "A survey on enhanced subspace clustering". In: *Data Mining and Knowledge Discovery* 26.2 (2013), pp. 332–397. DOI: 10.1007/s10618-012-0258-x.

[TSL00] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500 (2000), pp. 2319–2323. DOI: 10.1126/science.290.5500.2319.

[TXO05] A. K. H. Tung, X. Xu, and B. C. Ooi. "CURLER: Finding and Visualizing Nonlinear Correlation Clusters". In: *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. SIGMOD '05. Baltimore, Maryland: ACM, 2005, pp. 467–478. DOI: 10.1145/1066157.1066211.

[Tzo+02] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot. "Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain". In: *Neuroimage* 15.1 (Jan. 2002), pp. 273–289. DOI: 10.1006/nimg.2001.0978.

[VEB09] N. X. Vinh, J. Epps, and J. Bailey. "Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?" In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: ACM, 2009, pp. 1073–1080. DOI: 10.1145/1553374.1553511.

[WB68] C. S. Wallace and D. M. Boulton. "An Information Measure for Classification". In: *The Computer Journal* 11.2 (1968), pp. 185–194. DOI: 10.1093/comjnl/11.2.185.

[Wu+08] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. "Top 10 algorithms in data mining". In: *Knowledge and Information Systems* 14.1 (2008), pp. 1–37. DOI: 10.1007/s10115-007-0114-2.

[Yao03] Y. Y. Yao. "Information-Theoretic Measures for Knowledge Discovery and Data Mining". In: *Entropy Measures, Maximum Entropy Principle and Emerging Applications*. 2003, pp. 115–136. DOI: 10.1007/978-3-540-36212-8_6.

[YH02] X. Yan and J. Han. "gSpan: graph-based substructure pattern mining". In: *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on IS - SN - VO -*. 2002, pp. 721–724. DOI: 10.1109/ICDM.2002.1184038.

[YL12] J. Yang and J. Leskovec. "Community-Affiliation Graph Model for Overlapping Network Community Detection". In: *ICDM'12*. IEEE, Nov. 2012, pp. 1170–1175. DOI: 10.1109/ICDM.2012.139.

[YL13] J. Yang and J. Leskovec. "Overlapping Community Detection at Scale: a Nonnegative Matrix Factorization Approach." In: *WSDM* (2013), pp. 587–596. DOI: 10.1145/2433396.2433471.

[YT05] J. Yin and Z. Tan. "Clustering Mixed Type Attributes in Large Dataset". In: *Parallel and Distributed Processing and Applications*. Ed. by Y. Pan, D. Chen, M. Guo, J. Cao, and J. Dongarra. ISPA '05. Berlin, Heidelberg: Springer, 2005, pp. 655–661. DOI: 10.1007/11576235_66.

[YW06] Q. Yang and X. Wu. "10 Challenging Problems in Data Mining Research". In: *International Journal of Information Technology & Decision Making* 5.4 (2006), pp. 597–604. DOI: 10.1142/S0219622006002258.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases." In: SIGMOD '96. Montreal, Quebec, Canada: ACM, 1996, pp. 103–114. DOI: 10.1145/233269.233324.

List of Figures

List of Tables