Explaining and Visualizing Structural Knowledge in Bipartite Graphs

Dissertation

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt im

Fachbereich Elektrotechnik/Informatik

der

Universität Kassel

von

Dominik Dürrschnabel, M. Sc.

begutachtet durch

Prof. Dr. Gerd Stumme, Universität Kassel

Prof. Dr. Sebastian Rudolph, TU Dresden

vorgelegt am 28. Februar 2023

Disputation am 24. Mai 2023

Acknowledgements

Writing a doctoral thesis would not have been possible without a wide range of supporters that accompanied my journey. First and foremost, I am extremely grateful to my supervisor Prof. Dr. Gerd Stumme for his support in all my research interests, for his ideas when I was stuck or out of clues, and for providing an open environment.

Enjoying my time at the Knowledge and Data Engineering Group was also enabled by my colleagues who were always open to interesting discussions over a cup of coffee and at the talks in our weekly seminars. In particular, I would like to acknowledge Maximilian Felde, Tobias Hille, Johannes Hirth, Maren Koyda, Bastian Schäfermeier, and Andreas Schmidt for the time, we spent together at the department. The seminar talks were also attended by Prof. Dr. Bernhard Ganter who, with his fast understanding and sparking ideas, was particularly helpful to me. Special thanks in that regard in particular to Dr. Tom Hanika who was willing to stay with me at the institute till late in the night to revise publication drafts, and who always provided his knowledge about adjoining research areas to push me forward. I am also very grateful to my office colleague Maximilian Stubbemann who always had time to think with me about my current questions, no matter whether they were of scientific or programming nature. I was able to navigate the workdays at the university with ease thanks to Björn Fries who kept technical issues at bay, and Monika Vopicka who managed the bureaucracy and paperwork. Without their help, my workdays would have been much more difficult.

This whole thesis was proofread by Klaus Dürrschnabel, Rebekka Dürrschnabel, Simone Dürrschnabel, Janina Gärtner, and in parts by Monika Desombre, Tamar Mirbach, Sebastian Murgul, and Sebastian Plenz whose help in this regard I very much appreciate.

I am very lucky to be blessed with a great family that has supported me for as long as I can remember. In the last six years, I have received very special support from my partner Melis Kılıçaslan who I am always extremely grateful to have by my side.

Finally, I want to thank everybody who supported me in the last years but is not mentioned here explicitly.

Zusammenfassung

Bipartite Graphen ermöglichen die Modellierung von Beziehungen zwischen zwei Arten von Entitäten als Daten. Datensätze in dieser Form kommen in vielen Bereichen vor, beispielsweise in sozialen Netzwerken, in der Biologie oder auch in der Wirtschaft. Dies macht die Analyse von bipartiten Graphen, häufig zu einer Notwendigkeit. Die formale Begriffsanalyse ist ein Forschungsgebiet, welches die Analyse solcher bipartiten Graphen ermöglicht, indem die Daten in sogenannten Begriffen gruppiert werden. Diese Begriffe werden dann in der Form eines Verbandes geordnet.

In der vorliegenden Arbeit verfolgen wir mehrere Ansätze für die Entdeckung, Erklärung und Visualisierung von strukturellem Wissen in bipartiten Graphen. Dabei untersuchen wir Wissen, welches in der Form von leicht erklärbaren Teilstrukturen in bipartiten Datensätzen vorkommt und beantworten die Frage, wie einige dieser Substrukturen entdeckt werden können. Diese Teilstrukturen erlauben es einem Menschen, einen Einblick in die Informationen zu bekommen, welche den Daten zugrunde liegen. Zu diesem Zweck betrachten wir verschiedene Substrukturen wie Kontranominalskalen und ordinale Faktoren und erforschen Algorithmen, um diese zu berechnen. Darüber hinaus befassen wir uns mit dem Visualisierungsproblem, welches der formalen Begriffsanalyse inhärent ist. In der formalen Begriffsanalyse ist das primäre Werkzeug zur Visualisierung von Wissen eine Hierarchie von Begriffen, welche, obwohl sie eindeutig definiert ist, durch viele verschiedene Zeichnungen dargestellt werden kann. Diese Diagramme enthalten zwar dieselben Informationen, sind aber zu einem unterschiedlichen

Maße lesbar. Wir stellen zwei Methoden zur Berechnung von menschenlesbaren Zeichnungen vor. Während die erste der beiden Techniken vom Forschungsgebiet des automatischen Graphenzeichnens inspiriert ist, hängt die zweite mit den von uns zuvor durchgeführten strukturellen Untersuchungen zusammen. Schließlich versuchen wir, die Frage zu beantworten, ob es möglich ist, formale Begriffe so in niedrigdimensionale Vektorräume einzubetten, dass die sich daraus ergebenden Einbettungen für Aufgaben der formalen Begriffsanalyse verwendet werden können. Dazu stellen wir einen Ansatz für das Lernen des Hüllenoperators der formalen Begriffsanalyse vor. Wir benutzen dafür eine neuronale Netzwerkarchitektur, welche durch den word2vec-Ansatz, einem Verfahren aus der natürlichen Sprachverarbeitung, inspiriert ist. Wir demonstrieren das Potenzial, indem wir begriffliche Strukturen in den berechneten Einbettungen wiederzuentdecken.

Abstract

Bipartite graphs are an important model for the representation and analysis of relationships between two different types of entities. Datasets in this form are commonly found in many fields, such as social networks, biology, and economics. Formal concept analysis is a research approach that allows for the analysis of such bipartite graphs by clustering the data into so-called concepts and ordering those in a lattice structure.

In this thesis we propose multiple approaches for the extraction and visualization of structural knowledge from bipartite graphs. To this end, knowledge is presented as easily explainable substructures that appear in the dataset. We address the question, how to extract some of these substructures which provide insight into the information underlying the data. For this purpose, we consider several substructures such as contranominal scales and ordinal scales and provide algorithms on how to discover them. Furthermore, we tackle the visualization problem that is inherent to formal concept analysis. In formal concept analysis, the primary tool to visualize knowledge is the hierarchy of concepts which, even though it is unambiguously defined, can be represented by many different diagrams that entail the same information but are to a different degree readable. We provide two ways to compute human-readable drawings, one of them is inspired by techniques from graph drawing and the other is heavily related to the structural investigations that we performed before. Finally, we investigate the question, whether it is possible to embed formal concepts into a low-dimensional vector space to enhance tasks in formal concept analysis. Thereby, we present an algorithm for embedding formal concepts by learning the closure operator using a neural network architecture. This is inspired from the word2vec approach which was developed in the field of natural language processing. We demonstrate its potential in that regard as we are able to rediscover conceptual features in the computed embeddings.

Scientific Publications that Contribute to this Thesis

The following previously published works are incorporated into this thesis.

[28] Dominik Dürrschnabel, Tom Hanika, and Maximilian Stubbemann.
FCA2VEC: Embedding Techniques for Formal Concept Analysis.
In: Complex Data Analytics with Formal Concept Analysis (2022), pages 47–74.
DOI: 10.1007/978-3-030-93278-7_3. Reproduced with permission from Springer Nature.

This is a joint work with Maximilian Stubbemann and Tom Hanika and incorporated into Chapter 8. The research in this paper was mainly conducted by Maximilian Stubbemann and the author of this thesis under the advice of Tom Hanika. The related work in Chapter 8 is based on the joint work with Maximilian Stubbemann, the research part of this chapter is the original work of the author of this thesis.

[30] Dominik Dürrschnabel, Tom Hanika, and Gerd Stumme.
DimDraw - A Novel Tool for Drawing Concept Lattices.
In: Supplementary Proceedings of ICFCA 2019 Conference and Workshops, Frankfurt, Germany, June 25-28, 2019.

[31] Dominik Dürrschnabel, Tom Hanika, and Gerd Stumme.
Drawing Order Diagrams through Two-Dimension Extension.
`https://arxiv.org/abs/1906.06208`, submitted.

These two papers are incorporated into Chapter 7 and partly into Chapter 3. They were co-developed in many helpful discussions with Tom Hanika and Gerd Stumme.

[34] Dominik Dürrschnabel, Maren Koyda, and Gerd Stumme.
Attribute Selection using Contranominal Scales.
In: Graph-Based Representation and Reasoning - 26th International Conference on Conceptual Structures, ICCS 2021, Virtual Event, September 20-22, 2021, Proceedings. DOI: 10.1007/978-3-030-86982-3_10.

The research in the scientific publication was conducted by Maren Koyda and the author of this thesis and advised by Gerd Stumme. The parts that are incorporated into this thesis are in Chapter 4 and are original research from this author.

[36] Dominik Dürrschnabel and Gerd Stumme.
Force-Directed Layout of Order Diagrams using Dimensional Reduction.
In: Formal Concept Analysis - 16th International Conference, ICFCA 2021, Strasbourg, France, June 29 - July 2, 2021, Proceedings. DOI: 10.1007/978-3-030-77867-5_14.

Chapter 6 is taken from this publication. The publication was researched under the advice of Gerd Stumme.

[38] Dominik Dürrschnabel and Gerd Stumme.
Greedy Discovery of Ordinal Factors.
`https://arxiv.org/abs/2302.11554`, submitted.

[41] Dominik Dürrschnabel and Gerd Stumme.
Maximal Ordinal Two-Factorizations.
Accepted to: 28th International Conference on Conceptual Structures, ICCS 2023.

The research in these two publications was explored in discussions with Gerd Stumme and is integrated in Chapter 5.

Contents

Part I

Motivation and Foundations

CHAPTER 1

Introduction

Bipartite graphs provide a solid mathematical model for the representation of relationships between two groups of entities. In this section, we introduce challenges that emerge when extracting knowledge from such graphs and how we address them in this doctoral thesis. Thereby, we motivate the importance of knowledge discovery in data and the development of explainable methods as a whole. The main approach that we follow in this thesis to address this problem is the extraction of understandable substructures entailed in the datasets. Such patterns allow analysts to understand connections that are intrinsic to the dataset. Additionally, we use these patterns to generate readable order diagrams which allow the use of order as a structural paradigm for the investigation of complex hierarchical structures. In this thesis, we thus provide tools that support the understanding of relationships inherent to the data that can be used to perform informed decision-making.

1.1 Explaining and Visualizing Structural Knowledge in Bipartite Graphs

This thesis is located in the realm of formal concept analysis which is a subfield of mathematical data analysis. In data analysis, information is extracted from structured or unstructured data. Formal concept analysis which is a theory for structuring concepts in concept hierarchies can be used as a tool to discover knowledge, i.e., information that is useful and understandable in data that is modeled as bipartite graphs. In this thesis we contribute to the toolkit of formal concept analysis. To do so, we follow multiple research questions which can be condensed to one guiding research question:

How can the discovery of substructures in relational data enable humans to understand connections in the dataset?

This first problem that we work on is directly induced by the guiding question, as we cannot evaluate the usefulness of a substructure that we cannot compute. Thus, the following issue is of high relevance.

(1) How can we efficiently discover relevant substructures?

To this end, we are interested in discovering bipartite graphs in (non-bipartite) graphs, identifying contranominal scales in formal contexts, and discovering ordinal factors in formal contexts. The problem to discover bipartite graphs in graph data is re-appearing throughout this thesis as a supporting tool for the other problems and thus poses the following question.

(1.1) How can we discover bipartite graphs in non-bipartite graphs?

The second question in this regard is discovering contranominal scales, which are parts of a bipartite graph dataset that disproportionately contribute to the number of concepts, i.e., hierarchical elements in formal concept analysis, while providing little-to-none implicational knowledge.

(1.2) How can we discover contranominal scales in bipartite graphs?

Finally, we examine ordinal factors which are a valuable tool to examine relationships between different attributes.

(1.3) How can we discover ordinal factors in bipartite graphs?

We approach these three questions in the first part of the thesis. Each research question thereby corresponds to one of Chapters 3 to 5 respectively.

Order is the primary structure paradigm in formal concept analysis and order diagrams are the primary tool that allow humans to understand the relationships within an ordered set. While order is a well-defined concept in mathematics and the visualization technique of order diagrams is well-established, generating readable order diagrams remains an unsolved problem – there are no automatic techniques that come close to what an experienced human can create. This induces our next research question.

> **(2) How can we automatically generate human-readable diagrams of ordered sets?**

Two approaches come natural in this regard. As the order diagram problem can be seen as a special case of the graph drawing problem, we can tailor methods from this research realm to order diagram drawing.

> **(2.1) How can we adapt graph drawing techniques to order diagram drawing?**

The second approach to address this question is to leverage information encapsulated in the data for a more structurally motivated algorithm following the next question.

> **(2.2) How can we make use of structural properties of ordered sets to draw order diagrams?**

We address these two questions in Chapters 6 and 7.

There is a recent trend to employ vector space embeddings for all kinds of structural investigations, as those can be examined with the rich toolkit of linear algebra. We are interested, how algorithms in formal concept analysis could profit from such vector space embeddings. The question that poses itself in this regard thus links such embeddings to structural knowledge in the dataset.

> **(3) How can we leverage vector space embeddings of bipartite graphs to support knowledge discovery in formal concept analysis?**

We examine how an embedding approach using an architecture inspired by word2vec to embed the concepts into a vector space can help to achieve this in Chapter 8.

1.2 General Placement in Computer Science

In this section, we embed our research in the broader field of computer science by motivating some surrounding concepts. We strive to discover explainable structural knowledge from data sources which have the form of bipartite graphs. The first question that poses itself in this regard is, what we refer to when we talk about the notions of knowledge and explainability. We are going to introduce those in the following section. Subsequently, we will motivate why bipartite graphs provide an interesting structural basis, and why this thesis focuses on this data model. As this thesis is located in the realm of formal concept analysis, a theory whose main tool is a hierarchy of concepts, we are going to introduce the importance of order theory as a whole, and then we embed formal concept analysis into this context. Many of the algorithms and structures that emerge in formal concept analysis are of exponential nature. This results into several computational hurdles that we encounter in this thesis. We embed those into the context of computational complexity in this last part of this section, as many of the research questions formulated above entail NP-complete problems.

1.2.1 Knowledge and Explainability

One possible definition for knowledge discovery is due to Fayyad et al. [47].

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

In this definition Fayyad et al. determine four defining adjectives for the requested patters. Knowledge should be **valid**, i.e., the structure that we extract from the data should not contain false information, and it should be **novel** what implies that we are interested in information that is not directly apparent. Furthermore, it has to be **useful** and **understandable**. These two points determine that an analyst should be able to gain insight into the extracted structures, which is in contrast to many of the recent advances in data analysis. Especially neural networks notoriously struggle from a blackbox-character, i.e., they only present the generated information and do not explain how they arrived at the predictions, making it hardly possible for a human to understand the generated results. However, sensitive applications such as healthcare, finance, or justice necessitate trust in the results which makes explainability a prerequisite of utmost importance.

In this thesis we extract understandable and explainable structures, such as linear orders from the data when we approach the three subquestions of (1). Furthermore, the diagram-drawing algorithms that we introduce in relation to the question (2), allow a human reader to explore an order which entails the knowledge from the dataset. Even the vector space embedding which we propose when we consider question (3) can contribute to human understandability by being in low (in our case three) dimensions and allowing for the visualization of related concepts.

1.2.2 Bipartite Graphs

The research realm of graph theory was founded by Leonhard Euler in 1736 [46] when he solved the famous Seven Bridges of Königsberg problem. The question

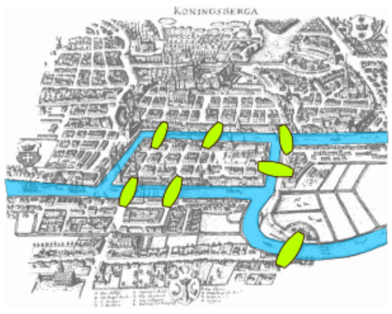


Figure 1.1: A map [111] of the historical Königsberg with the seven bridges. Euler proved that there is no tour that crosses each bridge exactly once and thereby started the research area of graph theory.

he considered was if it was possible to find a tour which crossed every bridge of historic Königsberg (compare to Figure 1.1) exactly once. He also asked whether one could return to the starting point without using a bridge twice. In this, Euler made the observation that the exact geometrical position of the bridges is not necessary to solve the problem. It is only important which parts of land they connect. One can argue that, when Euler showed that such a tour was not possible, he did not only introduce graph theory but was also the first person to extract knowledge from modeling a problem through a graph. Still, graph theory was after its introduction mainly used as a tool in the field of mathematics where it supported other areas such as topology and geometry, and provided ways to solve problems related to these fields. However, with the advent of computers in the mid-20th century, graph theory started to be applied to new emerging problems. Many of the data structures that modern computer science is built on are grounded in the area of graph theory. Furthermore, many algorithms that computer systems use such as routing algorithms are by the nature of the underlying data graph algorithms. Additionally, a large part of the recent advances in data analysis, machine learning, and artificial intelligence are due to the application of graphs.

Bipartite graphs are special graphs that can be used to model relationships between two different types of entities, such as people and events they attend, or actors that play in movies. They are commonly applied in data analysis and problem solving such as matching problems where a bipartite graph models the compatibility of entities between two sets and a valid assignment has to be found. Another research area that builds on bipartite graphs are recommender systems where recommended items for individual users are computed based on the bipartite graph of items which the user previously liked. Finally, the research realm of formal concept analysis, in which this thesis is to be located, makes use of bipartite graphs as its data source for knowledge discovery.

1.2.3 Order as a Structure Paradigm

Coombs [22] distinguishes two relations in which a pair of data points can stand to each other, namely *proximity* and *order*. The notion of proximity allows measuring whether two points are close, i.e., their distance is below a certain threshold, or far. Order on the other hand, which is of higher importance for this thesis, refers to a hierarchical relation between the data points. Usually, when we refer to order in an informal way, we assume that every pair of elements is comparable. In other words, we assume that either one element is “greater” than the other, or they are “equal”. However, in reality, we often encounter situations where not every pair of elements can be directly compared. An example for such a situation is a triathlon, where one athlete might be faster in swimming but slower in cycling than the other one. In this case, there is no emergent way to define the better athlete. Thus, the investigation of elements that are only partially ordered by a relation is of interest. This means that, for some pairs of elements, we can say that one is “greater” than the other, but for other pairs, there is no defined ordering. To make the two notions better distinguishable, we refer to the case where every pair of elements is comparable as a linear (or total) order.

1.2.4 Formal Concept Analysis

In this thesis, a special kind of ordered sets, so-called lattices, are of special interest. They are a key mathematical structure that underlie many algorithms in knowledge discovery, and they provide the structural basis for formal concept analysis (FCA). FCA is a mathematical theory that was founded by the mathematician Rudolf Wille in 1982 [113] as a theory for concepts and concept hierarchies based on lattice theory. It uses datasets consisting of objects and attributes together with an incidence relation that can be modeled as bipartite graphs as its data source. Subsets of the objects that share the same attributes in their incidence are clustered into so-called formal concepts which then constitute the order structure of a lattice. This lattice allows for the representation of the relationships between the objects and their attributes, and organize the data in a meaningful way. Object concepts are placed below attribute concepts, if and only if the object is incident to an attribute. Furthermore, so-called implications, i.e., relationships between the attributes, are represented in this concept lattice. Thus, formal concept analysis provides a way to extract relationships in data that may not be immediately apparent otherwise. The hierarchy of the concepts can be visualized for a human which allows for the analysis of complex data sets and the explanation of the inferred knowledge in a consistent and logical way.

1.2.5 Computational Obstacles

The computational complexity of a problem is a measure for the amount of resources, such as time and space that are required to solve it. It is a tool to identify the limits for solving a problem and is of high relevance and therefore has to be investigated. Understanding the limitations that are imposed through this complexity allows for the design of optimized algorithms and data structures.

The class of computational problems that is known as NP comprises all problems for which it is possible to verify a single solution efficiently. Efficiently in this context refers to polynomial time with respect to the size of the input. If not only a solution can be verified but also computed in efficient time, an algorithm is said to be in the problem class P. Surprisingly, it is still an open question, whether there are problems that are in NP but not in P. This is arguably the most important open problem in computer science whether the computational classes P and NP are equal. If this were the case, it would have profound implications on a wide range of fields. For example, most currently-used cryptographic methods could be easily broken. On the other hand, if $\mathsf{P} \neq \mathsf{NP}$, it would mean that the hardest problems in NP, the NP-complete problems, cannot be solved in polynomial time. In this case, the search for algorithms for NP-complete problems within the limits of this class becomes an even more important area, as those problems appear naturally in many contexts. The search for efficient algorithms for NP-complete problems is motivated by the quest to understand the limits of computation and the inherent difficulty of certain problems. Even if it might not be possible to solve NP-complete problems quickly, understanding the complexity of these problems and developing efficient algorithms or heuristics provides valuable insights into their nature.

In this thesis, we deal with the NP-complete problem on induced bipartite subgraphs in Chapter 3 when we approach question (1.1). To this end, we develop three heuristics that compute solutions for a weaker version of the problem. Similarly, in Chapter 4 we deal with the NP-complete problem of computing induced contranominal scales which can appear as subcontext of a formal context. In this chapter, we propose an exact algorithm which is coupled with several speed-up techniques. The problems that we consider in Chapter 5 can be linked to three distinct problems which are all NP-complete, as ordinal factorizations are closely related NP-complete order dimension problems. The same problems reappear in Chapter 7 as these two chapters share underlying theory.

1.3 Structure of this Thesis

The remainder of the current introductory part of this thesis introduces the mathematical principles and notations required for the rest of this work. After that, this thesis is divided into four parts, the first three of which deal with the research conducted.

In Part II, we approach the questions which relate to research question (1). In particular, for research question (1.1) we propose algorithms for the discovery of bipartite graphs in (non-bipartite) graphs in Chapter 3. To this end, we propose an exact solution of the global version of the problem leveraging fast SAT-solving algorithms and present three heuristics for a local version of the problem with different time to performance trade-offs. Then we propose an algorithm for the discovery of contranominal scales in Chapter 4 to address question (1.2). We equip this algorithm with several speed-up techniques. Finally, in Chapter 5, we propose two directions to compute ordinal factorizations with respect to research question (1.3). Thereby, we fill in the gap how to compute large ordinal two-factorizations for a given formal context and propose an algorithm to compute complete greedy factorizations.

Then in Part III, we approach the questions related to research question (2). To this end, considering question (2.1), we propose a force directed layout in Chapter 6. This is a visualization approach that uses physical principles such as attraction and repulsion, For research question (2.2), we propose an approach in Chapter 7 to compute visualizations based on the dimensional structure encapsulated into the data. Thereby, we make use of a bipartite subgraph that we compute using the techniques developed prior in this thesis.

Finally, in Part IV, we address question (3) and show some first promising results for the application of our vector space embeddings. We propose an embedding approach using an architecture that is inspired by word2vec to embed formal concepts into a vector space in Chapter 8. Using this embedding, we demonstrate that we are able to rediscover structural properties that stem from the conceptual structure. This allows us to analyze the relationships between different concepts in a more comprehensive and detailed manner. We are hopeful that, in the future, those approaches can support algorithms in formal concept analysis.

We conclude this thesis in Part V, where we summarize the results and give an outlook on potential follow-up research that can be conducted building on the work of this thesis.

CHAPTER 2

Mathematical Foundations

This thesis presents methods for data analysis and knowledge discovery which are rooted in mathematics. Here, especially the realm of graph theory is of special importance as it is used to model network structures. Graphs are also the foundation for the theory of formal concept analysis, a research area that investigates data in the form of bipartite graphs. This is a method that structures the data of a bipartite graph in an order structure called lattice. Order is a reoccurring notion that allows to compare individual entities or determine for entity pairs that they are incomparable. This thesis is founded in the area of graph theory, order theory, and formal concept analysis. We thus provide the required reoccurring mathematical terms for all three fields in this chapter. We restrict our repetition to the notions relevant for this work, the following recap should therefore not be considered complete.

2.1 Graph Theory

We keep the notations and notions in graph theory consistent to the ones in the standard book [25] written by Diestel. A commonly used shorthand notation for some $n \in \mathbb{N}$ is the set of natural numbers at most n given by $[n]$, i.e.,

$$[n] := \{i \in \mathbb{N} \mid i \leq n\} = \{1, 2, \dots, n\}.$$

For some set M, we adhere to the convention of denoting all subsets of cardinality two as

$$\binom{M}{2} := \{X \subseteq M \mid |X| = 2\}.$$

This notation now allows us to formally introduce a graph.

Definition 2.1 (Graph)
A *graph* is a tuple $G = (V, E)$ consisting of a set V called *vertices* and a set $E \subseteq \binom{V}{2}$ called *edges*. We say two vertices $u, v \in V$ are *adjacent* if $\{u, v\} \in E$. In this case, v is called a *neighbor* of u. The degree of a vertex u is the number of its neighbors and denoted by $\deg(u)$. We call $|V|$ the *order* and $|E|$ the *size* of a graph.

An example for a graph is given as follows.

Example 2.2 (Graph)
Let $G = (V, E)$ be a graph with the vertex set $V = \{a, b, c, d\}$ and the edge set $E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}\}$.

It should be noted that this is not the only way to define graphs. In some conventions, the edges of graphs are assigned with directions or multiple edges between the same vertices are allowed. Also, sometimes edges between a single vertex or more than two vertices are possible. To avoid ambiguities, the kind of graphs used in this work are in some instances called *simple undirected graphs without loops*, while edges with directions can be referred to as *digraphs*.

A way to geometrically present graphs are graph drawings which visualize the vertices and edges and allow an experienced reader to grasp the relationship in a graph quickly. Thereby, we require the notion of *Jordan curves*, which are defined as the image of a continuos and injective function that maps a closed interval into Euclidean space.

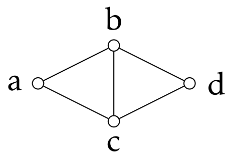


Figure 2.1: A drawing (or diagram) of the graph described in Example 2.2.

Definition 2.3 (Drawing)
A *graph drawing* or a *graph diagram* is a representation of a graph in the two-dimensional Euclidean space where each vertex is represented by a dot. Two dots are connected through a Jordan curves if and only if their corresponding vertices share an edge.

It should be noted that in a graph diagram for the exact representation of the graph the dots of two vertices are not allowed to touch. Similarly, it is not forbidden for a vertex v to touch an edge e if $v \notin e$. A possible graph drawing for the graph from Example 2.2 is presented in Figure 2.1. By definition, drawings of graphs are not unique and there is no straightforward way how to position the vertices or edges. A common convention for drawings is to visualize the edges if possible by straight lines, but it is also possible to use visualizations where the edges have bends or curves. A graph drawing provides a convenient way to visualize the structure of a graph as it is always possible to reconstruct the graph from its drawing.

In this thesis, we often perform structural investigations of graphs. If we rename the vertices, the structure of the graphs does not change, as long as we relabel the edges accordingly. It is thus of use to define a notion for graphs to be structurally equivalent.

Definition 2.4 (Isomorphisms of Graphs)
Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs. A bijective map $\varphi : V_1 \to V_2$ between G_1 and G_2 is called an *isomorphism* if for all elements $u, v \in V_1$ it holds that $\{u, v\} \in E_1$ if and only if $\{\varphi(u), \varphi(v)\} \in E_2$. Two graphs are called *isomorphic* if there is an isomorphism between them.

In Part II, we deal with substructures of graphs and formal contexts. It is therefore necessary to formally define substructures of graphs. There are two different notions of subgraphs, in both a subset of the vertices is selected. If the subgraph is not qualified further, it is allowed to pick an arbitrary subset of edges between the selected vertex set while for an induced subgraph all possible edges have to be included.

Definition 2.5 ((Induced) Subgraph)
Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. We call G_2 a *subgraph* of G_1 if $V_2 \subseteq V_1$ and $E_2 \subseteq E_1$. We call a subgraph *induced* if $E_2 = E_1 \cap \binom{V_2}{2}$. In this case, we denote $G_1[V_2] := G_2$.

It is common to refer to a graph H as a subgraph of another graph G if there is a subgraph G_2 of G that is isomorphic to H, even if they do not share any vertices or edges. The same notion also applies to induced subgraphs.

Some graphs possess uniquely defined characteristics, such as being *bipartite* or *complete*, which are often designated as qualifiers. A complete graph, for instance, comprises every possible edge and can be formally defined as follows:

Definition 2.6 (Complete Graph)
A *complete graph on $n \in \mathbb{N}$ vertices* is a graph (V, E) where $E = \binom{V}{2}$ and $n = |V|$.

Since this thesis is founded in the research area of formal concept analysis, the bipartite property is a particularly significant aspect.

Definition 2.7 (Bipartite Graph)
A graph $G = (V, E)$ is called *bipartite* if there are two sets A and B such that $A \cap B = \emptyset$, $A \cup B = V$ and $\binom{A}{2} \cap E = \binom{B}{2} \cap E = \emptyset$. In this case, A and B are called *bipartition classes*.

Our definition of a bipartite graph allows that the sets A and B can be empty. This means that a graph with just a single vertex or no vertices is considered bipartite. It should be noted that there are conventions which do not allow this. Still, these graphs are very special corner cases and of no particular interest.

Example 2.8 (Bipartite Graph)
The graph $G = (V, E)$ with the vertices $V = \{a, b, c, d, e, f\}$ and the edge set $E = \{\{a, d\}, \{a, f\}, \{b, e\}, \{b, f\}, \{c, d\}\{c, f\}\}$ is bipartite with the bipartition classes $A = \{a, b, c\}$ and $B = \{d, e, f\}$.

The drawing of the bipartite graph from Example 2.8 is depicted in Figure 2.2. Thereby, the bipartition classes are depicted by the dashed lines.

Definition 2.9 (Path)
Let (V, E) be a graph. A subgraph P on $n + 1$ distinct vertices $\{v_1, v_2, \ldots, v_{n+1}\} \subseteq V$ with the edge set $\{\{v_i, v_{i+1}\} \mid i \in [n]\} \subseteq E$ is called a path. We refer to P as a path of *length* n that connects v_1 to v_{n+1}.

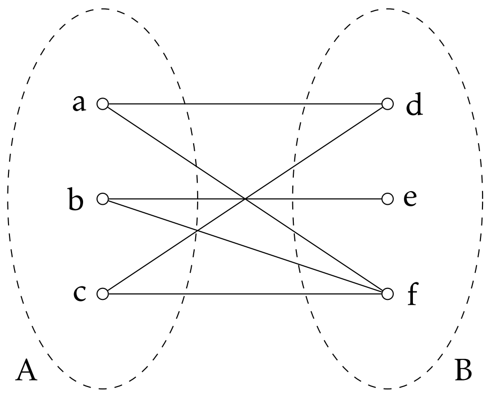


Figure 2.2: A drawing of the graph described in Example 2.8. The bipartition classes A and B are highlighted by the dashed lines.

In some notions, the length of a path is defined by the number of its vertices instead of the edges. To align this definition with our notion, we need to account for the discrepancy as a path has one additional vertex in comparison to the number of edges. The concept of a path in a graph can be used to determine whether it is connected.

Definition 2.10 (Connected Graph)
Let $G = (V, E)$ be a graph. G is called *connected* if for each pair of vertices $u, v \in V$, there is path connecting them.

Another graph of high interest is the cycle graph, as it can be used to characterize bipartite graphs.

Definition 2.11 (Cycle Graph)
A graph $G = (V, E)$ is called a cycle of length $|V|$ if it is connected and every vertex has degree two.

There are multiple equivalent ways to define a cycle graph. We could for example define it by the fact that for each pair of vertices $u, v \in V$, there have to be exactly two paths connecting them. It would also be possible to take a path of length $n - 1$ and connect the first and last vertex by an edge to achieve a definition of a cycle graph. In contrast to paths, it does not matter whether we count the number of vertices or edges to define the length of a cycle, as they are equal. A well known theorem from graph theory relates cycles of odd length to their bipartiteness as follows.

Theorem 2.12 (Characterization of Bipartite Graphs)
A graph is bipartite if and only if it does not contain a cycle of odd length as a subgraph.

2.2 Order Theory

The notations of order theory employed in this work align with those found in Trotter's reference book [106].

Definition 2.13 (Binary Relation)
A *binary relation* R between two sets A and B is given as a subset of the set $\{(a,b) \mid a \in A, b \in B\}$. It is denoted by $R \subseteq (A \times B)$. The shorthand notation for $(a,b) \in R$ is given by aRb and for $(a,b) \notin R$ by $a\not{R}b$. For the relation R we call $R^{-1} := \{(a,b) \mid (b,a) \in \mathcal{R}\}$ the *inverse relation* of R.

The focus of this thesis is on ordered sets which are a specific type of binary relation on some ground set. They are defined as follows.

Definition 2.14 (Ordered Set)
A binary relation $\leq \subseteq (X \times X)$ on a set X is called an *order relation* if and only if it satisfies the following properties:

1. *Reflexivity:* $\forall x \in X : x \leq x$.
2. *Antisymmetry:* $\forall x, y \in X : x \leq y \wedge y \leq x \Rightarrow x = y$.
3. *Transitivity:* $\forall x, y, z \in X : x \leq y \wedge y \leq z \Rightarrow x \leq z$.

The pair $(X, \leq)$ is called an *ordered set*.

The following is an example of an ordered set.

Example 2.15 (Ordered Set)
Let $X = \{a,b,c,d,e,f\}$ and $\leq = \{(a,a),(a,b),(a,c),(a,d),(a,e),(a,f),(b,b),(b,d),(b,e),$ $(b,f),(c,c),(c,d),(c,e),(c,f),(d,d),(d,f),(e,e),(e,f),(f,f)\}$. Then $(X, \leq)$ is an ordered set.

Strict orders are closely related to orders, with the difference being that strict orders are irreflexive instead of reflexive, i.e., all elements are not comparable to themselves, and the antisymmetry is replaced with asymmetry. For asymmetry if a pair is in the relation, its reversed pair cannot be in the relation. For antisymmetry on the other hand, this is possible and in this case the equality of both elements in the pair holds.

Definition 2.16 (Strictly Ordered Set)
A relation $< \subseteq (X \times X)$ where X is a set is called a *strict order relation*, if it satisfies the following properties:

1. *Irreflexivity:* $\forall x \in X : x \not< x$.
2. *Asymmetry:* $\forall x, y \in X : x < y \Rightarrow y \not< x$.
3. *Transitivity:* $\forall x, y, z \in X : x < y \wedge y < z \Rightarrow x < z$.

The pair $(X, <)$ is called a *strictly ordered set*.

In the above definition of a strictly ordered set, one could omit either irreflexivity or asymmetry, as either of these conditions together with the transitivity implies the other one. The relationship between strictly ordered set and ordered set is that they can be converted into each other. To convert an ordered set into a strictly ordered set, we can simply omit the reflexive pairs. On the other hand, adding reflexive pairs to a strictly ordered set results in the corresponding (not-strictly) ordered set. It is thus sensible for an ordered set $(X, \leq)$ and two pairs $a, b \in X$ to refer to the elements a to be strictly less than be if $a \leq b$ and $a \neq b$. In this case, we also use the notation $a < b$. Similarly, we say that b is strictly greater than a and denote it by $b > a$.

Example 2.17 (Strictly Ordered Set)
The strictly ordered set which corresponds to Example 2.15 is given by $(X, <)$ with $X = \{a, b, c, d, e, f\}$ and $< = \{(a,b), (a,c), (a,d), (a,e), (a,f), (b,d), (b,e), (b,f), (c,d), (c,e), (c,f), (d,f), (e,f)\}$.

On the other hand, if we remove the transitive and reflexive pairs from an ordered set, we obtain its covering relation.

Definition 2.18 (Covering Relation)
The *covering relation* of an ordered set $(X, \leq)$ is the subset of $\leq$ denoted by $\prec$ with

$$x \prec y :\Longleftrightarrow x < y \wedge \nexists z \in X : x < z < y.$$

Example 2.19 (Covering Relation)
The covering relation of the ordered set from Example 2.15 is given by the relation $\leq = \{(a,b), (a,c), (b,d), (b,e), (c,d), (c,e), (d,f), (e,f)\}$.

To compute the covering relation, we calculate the transitive reduction. On the other hand, the transitive closure, which will be defined next, can be used to determine all element pairs of the order based on the covering relation.

Definition 2.20 (Transitive Closure)
For a binary relation R on a set X, the *transitive closure* is defined as the minimal transitive superset of R and is denoted by R^+.

The transitive closure of an order relation can be computed by identifying and adding all missing transitive pairs to the covering relation, which can be done in $\mathcal{O}(|X|^3)$ with respect to the cardinality of the ground set X [49, 98, 110]. This process is the opposite of computing the transitive reduction, which involves finding the covering relation from the set of all pairs of an ordered set.

It is sensible to define a symmetric relation between a pair of elements to analyze, whether they are comparable.

Definition 2.21 (Comparability)
We say that two elements $a, b \in X$ are *comparable* if $a \leq b$ or $b \leq a$, otherwise they are *incomparable*.

Comparability gives rise to two associated graphs that are important for this thesis.

Definition 2.22 (Comparability and Cocomparability Graph)
For an ordered set $(X, \leq)$, its *comparability graph* is defined as the graph (X, E), such that $\{a, b\} \in E$ if and only if $a, b \in X$ are comparable. Similarly, the *cocomparability graph* (sometimes called *incomparability graph*) is the graph on X where $\{a, b\}$ is an edge if and only if $a, b \in X$ are incomparable. Two order relations on the same ground set are called *conjugate* to each other if the comparability graph of one is the cocomparability graph of the other.

If all pairs are comparable in an ordered set, it is usually referred to as one-dimensional or linear.

Definition 2.23 (Linear Order)
A (strictly) ordered set is called *linear* if every pair of elements is comparable.

Linear orders can either be strict or not strict. It is common to specify which type of linear order is being referred to, as both notions are frequently used. The concepts of suprema and infima are of special structural importance for this thesis.

Definition 2.24 (Supremum and Infimum)
Let $(X, \leq)$ be an ordered set with elements $a, b \in X$. An element s is called *supremum* of a and b if $a \leq s$ and $b \leq s$, there is no element $s' < s$ with $a \leq s'$ and $b \leq s'$, and s

is unique with this property. Dually, an element i is called *infimum* of a and b if $i \leq a$ and $i \leq b$, there is no element $i < i'$ with $i' \leq a$ and $i' \leq b$, and i is unique with this property.

The existence of unique suprema and infima for all pairs gives rise to the notion of lattices.

Definition 2.25 (Lattice)
An order $(X, \leq)$ is called a *lattice* if and only if for each pair $(a,b) \in X$, there is an infimum i and a supremum s. We denote s as $\sup(a,b)$ and i as $\inf(a,b)$. Additionally, we define the two binary commutative and associative operations $\vee$ and $\wedge$ on X with $a \vee b := \sup(a,b)$ and $a \wedge b := \inf(a,b)$.

Note that Example 2.15 is no lattice as the elements b and c do not have a unique supremum. We can also define a lattice without using the notion of order as follows.

Theorem 2.26 (Lattice Characterization)
A lattice is an algebraic structure $(X, \vee, \wedge)$ consisting of a set X and two binary, commutative and associative operators satisfying the identities

$$\begin{aligned} a \vee (a \wedge b) &= a, \\ a \wedge (a \vee b) &= a \end{aligned}$$

for all $a, b \in X$.

The structure of the lattice definition directly gives rise to the identities in the characterization. On the other hand, to achieve the definition from the characterization, one can define the order $\leq$ on the set X by $a \leq b$ if $a = a \wedge b$ or by $b = a \vee b$ for all elements $a, b \in X$. A special property of lattices that is of interest for data analysis is the notion of completeness.

Definition 2.27 (Complete Lattice)
A *complete lattice* is a lattice, in which all subsets of the ground set have a unique supremum and infimum.

It is important to mention at this point that if the ground set is finite, every lattice is complete.

Similar to graph drawings, there is a visualization method for ordered sets called order diagrams.

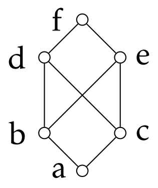


Figure 2.3: A diagram of the ordered set in Example 2.15. The lines represent exactly the covering relation. It is no lattice, as the elements b and c have no unique supremum.

Definition 2.28 (Order Diagram)
An *order diagram* of an ordered set $(X, \leq)$ is an embedding of the two-dimensional Euclidean space where each element of the order is represented by a dot. Two dots are connected by a monotonically increasing Jordan curve if they are in covering relation and for each pair of elements x, y with $x \leq y$, the vertical position of x is below y.

To ensure proper placement, no two elements may occupy the same coordinates and the Jordran curves may only touch the elements that are in covering relation. A linear extension of an order is a linear order that is consistent with the ordered set.

Definition 2.29 (Linear Extension)
For $(X, \leq)$, the order relation L on X is called a *linear extension* of $\leq$ if and only if L is a linear order and $\leq \subseteq L$.

The fewest possible number of linear orders to fully describe an ordered set leads to the notion of order dimension.

Definition 2.30 (Order Dimension)
For some $(X, \leq)$, let $\mathcal{R}$ be a family of linear extensions of $\leq$ such that $\leq = \bigcap_{L \subseteq \mathcal{R}} L$. We call $\mathcal{R}$ a *realizer* of $\leq$ of size d where d is the cardinality of a realizer. The minimal value of d such that there is a realizer of size d is called the *order dimension* of $(X, \leq)$.

We use the denotation of order dimension for ordered sets and their order relations interchangeably. By definition, a linear order is an order of order dimension one. If a linear order appears as subset of an ordered set, we call it a chain. The formal definition for a chain is given as follows.

Definition 2.31
Let $(X,\leq)$ be an ordered set. A subset of X where all elements are pairwise comparable is called a *chain*.

2.3 Formal Concept Analysis

This thesis is grounded in the field of formal concept analysis and utilizes the notations and concepts established in this area of research. To ensure compatibility with previous works in the field, we have adopted the notations presented in the standard reference for formal concept analysis [57]. We begin by defining formal contexts, a useful tool for representing binary datasets which we frequently encounter in this thesis.

Definition 2.32 (Formal Context)
For two sets G and M with an *incidence relation* $I \subseteq G \times M$, we call the triple (G,M,I) a *formal context*. The set G is thereby referred to as the set of *objects* while M is the set of *attributes*.

Note that if A and B are disjoint, a binary relation corresponds to a bipartite graph with A and B being the bipartition classes and I the edge set.

Throughout this thesis, we primarily focus on the finite case of G, M, and I as we are analyzing real world data. However, we also acknowledge that these sets can potentially be infinite and specifically mention any results that hold true for the infinite case.

Example 2.33 (Formal Context)
A formal context can be represented by a table where the objects are described by the columns names and the rows depict the attributes. Each entry in the table has a cross, if the corresponding object and attribute are incident. A cross table of a formal context depicting organizations and trieties of some select countries is given in Figure 2.4.

Sometimes, it is useful to explore the opposite perspective, where the incidence relation reflects which attributes an object lacks.

Definition 2.34 (Complementary Context)
For a formal context $\mathbb{K} = (G,M,I)$, let the *complementary formal context* be the context $\mathbb{K}^c := (G,M,(G\times M)\setminus I)$.

	Council of Europe	EFTA	EU	Euro	NATO	OSCE	Schengen
Austria	×		×	×		×	×
Belgium	×		×	×	×	×	×
Czech Republic	×		×		×	×	×
Denmark	×		×		×	×	×
France	×		×	×	×	×	×
Germany	×		×	×	×	×	×
Lexembourg	×		×	×	×	×	×
Netherlands	×		×	×	×	×	×
Poland	×		×		×	×	×
Switzerland	×	×				×	×

Figure 2.4: A formal context representing the countries bordering Germany with some treaties and organizations in which they partake.

In formal concept analysis, there are two derivation operators that are used to form a connection between attributes and objects through the incidence relation.

Definition 2.35 (Derivation Operators)
For a formal context $\mathbb{K} = (G, M, I)$, the *derivation operators* are defined as follows

$$\begin{aligned} A' &= \{m \in M \mid \forall g \in A : (g,m) \in I\}, \\ B' &= \{g \in G \mid \forall m \in B : (g,m) \in I\}. \end{aligned}$$

We denote for a single attribute m the derivation $m' := \{m\}'$ and for a single object g we write $g' := \{g\}'$.

Here, we follow the convention to use the same notation for attribute and object derivations. While it is common for attributes and objects to be distinct sets, it is possible for them to overlap. In these cases, the derivation operators must be specified in order to make them distinguishable. However, in most cases, the context makes it clear which derivation is being applied.

The derivation operators are well investigated and are known to comply with the following properties.

Theorem 2.36 (Properties of the Derivation Operators)
Let (G,M,I) be a formal context. Then

(1) $\forall A_1,A_2 \subseteq G : A_1 \subseteq A_2 \Rightarrow A_2' \subseteq A_1'$, *(4)* $\forall B_1,B_2 \subseteq M : B_1 \subseteq B_2 \Rightarrow B_2' \subseteq B_1'$,
(2) $\forall A \subseteq G : A \subseteq A''$, *(5)* $\forall B \subseteq M : B \subseteq B''$,
(3) $\forall A \subseteq G : A' = A'''$, *(6)* $\forall B \subseteq M : B' = B'''$,
(7) $\forall A \subseteq G, B \subseteq M : A \subseteq B' \iff B \subseteq A'$.

Note that properties *(1)*, *(2)*, and *(3)* are dual to properties *(4)*, *(5)*, and *(6)* respectiviely. The derivation operator give rise to two closure operators, which are defined as follows.

Definition 2.37 (Closure Operator)
Let X be a set. An operator $\mathrm{cl} : \mathcal{P}(X) \to \mathcal{P}(X)$ on the power set of X is called a *closure operator*, if it has the following properties.

1. *Extensity:* $\forall A \subseteq X : A \subseteq \mathrm{cl}(A)$
2. *Monotony:* $\forall A,B \subseteq X : A \subseteq B \Rightarrow \mathrm{cl}(A) \subseteq \mathrm{cl}(B)$
3. *Idempotency:* $\forall A \subseteq X : \mathrm{cl}(\mathrm{cl}(A)) = \mathrm{cl}(A)$

Because of the properties from above, it follows that the derivation operators from formal concept analysis give rise to two closure operators as follows.

Theorem 2.38 (Closure Operators of Formal Concept Analysis)
Let (G,M,I) be a formal context the operators

$$\mathcal{P}(G) \to \mathcal{P}(G) : (\cdot) \mapsto (\cdot)''$$

and

$$\mathcal{P}(M) \to \mathcal{P}(M) : (\cdot) \mapsto (\cdot)''$$

are closure operators.

As the derivation operators have these closure-property, they can be used for clustering the attribute and object sets into useful concepts.

Definition 2.39 (Formal Concept)
For a formal context $\mathbb{K} = (G,M,I)$, we call all tuples (A,B) with $A \subseteq G$ and $B \subseteq M$ such that $A' = B$ and $B' = A$ the *formal concepts*. The set A is called the *extent* of the formal concept while the set B is called the *intent*. We denote the set of all formal concepts as $\mathfrak{B}(G,M,I)$ or if the context is unambigous as $\mathfrak{B}$.

To illustrate the concepts of a formal context, consider once again the example from above.

Example 2.40 (Formal Concept)
The set of all concepts for the context from Example 2.33 is given by

- ((Austria, Belgium, Czech Republic, Denmark, France, Germany, Lexembourg, Netherlands, Poland, Switzerland), (Council of Europe, OSCE, Schengen)),
- ((Austria, Belgium, Czech Republic, Denmark, France, Germany, Lexembourg, Netherlands, Poland), (Council of Europe, EU, OSCE, Schengen)),
- ((Belgium, France, Germany, Lexembourg, Netherlands), (Council of Europe, EU, Euro, NATO, OSCE, Schengen)),
- ((Austria, Belgium, France, Germany, Lexembourg, Netherlands), (Council of Europe, EU, Euro, OSCE, Schengen)),
- ((Belgium, Czech Republic, Denmark, France, Germany, Lexembourg, Netherlands, Poland), (Council of Europe, EU, NATO, OSCE, Schengen)),
- ((Switzerland), (Council of Europe, EFTA, OSCE, Schengen)),
- ((), (Council of Europe, EFTA, EU, Euro, NATO, OSCE, Schengen).

The central theorem that allows formal concept analysis to cluster the concepts in a hierarchy is the basic theorem of formal concept analysis.

Theorem 2.41 (The Basic Theorem on Concept Lattices)
The tuple $\underline{\mathfrak{B}}(G,M,I) = (\mathfrak{B}, \leq)$ with $(A_1,B_1) \leq (A_2,B_2)$ if $A_1 \subseteq A_2$ gives rise to a complete lattice in which infimum and supremum are given by

$$\bigwedge_{t\in T}(A_t,B_t) = \left(\bigcap_{t\in T} A_t, \left(\bigcup_{t\in T} B_t\right)''\right),$$
$$\bigvee_{t\in T}(A_t,B_t) = \left(\left(\bigcup_{t\in T} A_t\right)'', \bigcap_{t\in T} B_t\right).$$

A complete lattice $(L,\leq)$ is isomorphic to $\underline{\mathfrak{B}}(G,M,I)$ if and only if there exist two maps $\gamma : G \to L$ and $\mu : M \to L$ such that every element of L can be constituted by the supremum of a subset of $\gamma(G)$, and by the infimum of a subset of $\mu(M)$, and $(g,m) \in I$ if and only if $\gamma(g) \leq \mu(m)$ for all $g \in G$ and for all $m \in M$. In particular $(L,\leq)$ is isomorphic to $\underline{\mathfrak{B}}(L,L,\leq)$

This implies that the set of all formal concepts can be ordered in a lattice structure. We refer to the complete lattice that is defined by the basic theorem as the *concept*

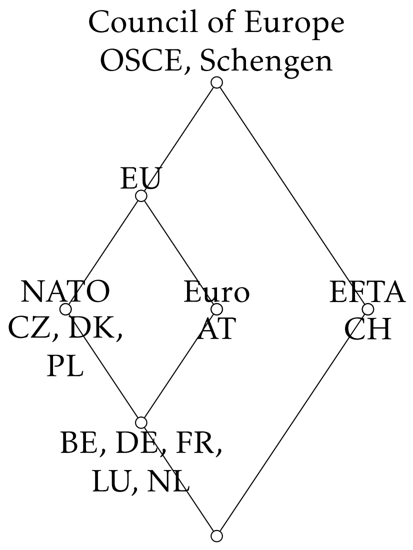


Figure 2.5: The concept lattice with labels of the formal context from Example 2.33.

lattice. If $\mathbb{K} = (G,M,I)$ we refer to $\underline{\mathfrak{B}}(G,M,I)$ for short as $\underline{\mathfrak{B}}(\mathbb{K})$. In a drawing of the order diagram of such a concept lattice, we can label the elements as follows. The largest concept that contains some attribute gets the attribute label placed above. Dually, the smallest concept that contains some object gets annotated with the object name from below. Thanks to the basic theorem above, this allows the deduction of all attributes and objects in a concept from the drawing. A concept exactly has an attribute if there is a larger concept with the attribute label. Dually, a concept exactly contans an object if there is a smaller concept with the object label. Consider the following as an exemplification.

Example 2.42 (Concept Lattice)
The concept lattice from Example 2.33 is visualized in Figure 2.5.

The computation of the set of all formal concepts can be done using algorithms such as the famous NEXTCLOSURE [55] algorithm or the TITANIC [103] algorithm. Both these algorithms can compute a lattice of all closures for an arbitrary given closure operator.

For a given formal context, it is possible to capture implications between the attributes, which formalizes the statement "Every object with the attributes $a,b,c,\dots$ also has attributes $x,y,z,\dots$".

Definition 2.43
Let $\mathbb{K} = (G, M, I)$ be a formal context and $A, B \subseteq M$. We say that A implies B in $\mathbb{K}$ if $A' \subseteq B'$ (or $B \subseteq A''$). In this case, we call $A \to B$ an *implication* of $\mathbb{K}$.

There are algorithms for the computation of the set of all implications and algorithms that directly compute a minimal base. As this is not of relevance for this thesis, we will not go into deeper detail.

Similarly to subgraphs, it is possible to define substructures of formal contexts.

Definition 2.44 ((Induced) Subcontext)
Let $\mathbb{K} = (G, M, I)$ be a formal context. A formal context (H, N, J) is called a *subcontext* of $\mathbb{K}$ if $H \subseteq G$, $N \subseteq M$, and $J \subseteq I$. It is called an *induced subcontext* if $J = I \cap (H \times N)$ and denoted by $\mathbb{K}[H, N]$.

By clarifying and reducing a formal context, we can achieve a formal context with less attributes, objects and incidence pairs without changing the structure of its concept lattice.

Definition 2.45 (Clarifying and Reducing)
Let $\mathbb{K} = (G, M, I)$. We call an attribute m *clarifiable* if there is an attribute $n \neq m$ with $n' = m'$. In addition, we call it *reducible* if there is a set $X \subseteq M$ with $m \notin X$ and $m' = X'$. Otherwise, we call m *irreducible*. $\mathbb{K}$ is called *attribute clarified* (*attribute reduced*) if it does not contain clarifiable (reducible) attributes. The definitions for the object set are analogous. If $\mathbb{K}$ is attribute clarified and object clarified (attribute reduced and object reduced), we say $\mathbb{K}$ is *clarified* (*reduced*).

Every formal context $\mathbb{K}$ has a unique reduced (or clarified) subcontext up to isomorphism. Furthermore, both their concept lattices are isomorphic to $\underline{\mathfrak{B}}(\mathbb{K})$. Because of the duality between attributes and objects, everything discussed in this section can be considered in a dual setting by interchanging attributes and objects.

2.3.1 Special Formal Contexts

In this section, we introduce formal contexts with their concept lattices that are of special interest. Those are reoccurring in different sections of this work. The ordinal scale is the formal context that gives rise to linear orders.

	1	2	3	$\cdots$
1	$\times$			$\cdots$
2	$\times$	$\times$		$\cdots$
3	$\times$	$\times$	$\times$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

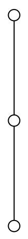


Figure 2.6: *Left:* The formal context of an ordinal scale. *Right:* The concept lattice of an ordinal scale of length 3.

Definition 2.46
For $n \in \mathbb{N}$, we call the formal context $([n],[n],\leq)$ an *ordinal scale* of *length n*.

All concepts in an ordinal scale are of the form $([n] \setminus [i-1],[i])$ for $i \in \{1,2,\ldots,n\}$. Thus, the ordinal scale of length n contains n concepts. All these concepts are comparable and therefore the concept lattice of the ordinal scale forms a linear order. An example for an ordinal scale is given in Figure 2.6.

While the ordinal scale gives rise to a concept lattice of order dimension one, the contranominal scale is the formal context that generates the highest possible order dimension that can be attained with its number of attributes and objects.

Definition 2.47
For a natural number n the formal context $([n],[n],\neq)$ is called a *contranominal scale* of *dimension n*.

The concept lattice of the contranominal scale is called a *Boolean lattice of dimension* k and consists of 2^k concepts. It has order dimension k. An example is given in Figure 2.7.

The formal contexts introduced in this section are not complete in regard to the reoccurring ones in the realm of formal contexts. For the purposes of this thesis, we limit ourselves to the subcontexts defined above, as only these are relevant to our discussion.

	1	2	3	$\cdots$
1		$\times$	$\times$	$\cdots$
2	$\times$		$\times$	$\cdots$
3	$\times$	$\times$		$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

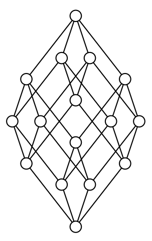


Figure 2.7: *Left:* The formal context called contranominal scale. *Right:* The concept lattice of the contranominal scale of dimension 3.

Part II

Discovering Substructures in Relational Data

CHAPTER 3

Induced Bipartite Subgraphs

Analyzing induced bipartite subgraphs of maximal vertex cardinality is an essential concept for the analysis of graphs. Yet, discovering them in large graphs is known to be computationally hard. This comes back to our first research question that we formulated at the beginning of this thesis.

(1.1) How can we discover bipartite graphs in non-bipartite graphs?

In this chapter, we consider two versions of this problem: the global problem and a weaker notion of it, where we discard the maximality constraint in favor of inclusion maximality. For solving the global version of the problem, we perform a reduction to a SAT problem and then employ the algorithmic strength of fast SAT-solvers. In the weaker notion, we aim to discover locally maximal bipartite subgraphs. For this, we present three heuristic approaches to extract such subgraphs and compare their results to the solutions of the global problem. Our three proposed heuristics are based on a greedy strategy, a simulated annealing approach, and a genetic algorithm, respectively. We evaluate all four algorithms with respect to their time requirement and the vertex cardinality of the discovered bipartite subgraphs on several benchmark datasets.

3.1 Introduction

Graphs are used in a variety of sciences to model and to analyze complex relationships. In this context, the search for interesting and relevant substructures is a standard procedure. Bipartite graphs represent a particularly interesting substructure since they allow a multitude of further mathematically intrinsic as well as scientifically extrinsic interpretations, e.g., communities in social networks or codewords in coding theory. Yet, discovering the largest bipartite subgraph of a graph is a problem that is known to be of high computational complexity. The central problem that we deal with in this chapter is the following.

Problem 3.1 (Global Induced Bipartite Subgraph Problem)
Given: A graph $G = (V, E)$.
Requested: A subset of vertices $X \subseteq V$ such that the graph $G[X]$ is bipartite and the cardinality of X is maximal.

As any choice of two vertices gives rise to an induced bipartite subgraph, this question is indeed well-defined. Still, it follows from a result by Lewis and Yannakakis [85] that finding such a set of maximal cardinality is computationally expensive. That is, deciding for a graph G whether a set of cardinality k exists that induces a bipartite subgraph is an NP-complete task. While the computational complexity of this problem is well-investigated, it is not yet explored how to efficiently discover large, yet not maximal, induced bipartite subgraphs of a given graph. This is the weaker notion that we also address in this section.

Problem 3.2 (Local Induced Bipartite Subgraph Problem)
Given: A graph $G = (V, E)$.
Requested: A subset of vertices $X \subseteq V$ such that the graph $G[X]$ is bipartite and there is no set $\tilde{X} \supsetneq X$ with the same property.
Optimization: Maximize $|X|$.

In this setting, we discard the requirement of X to be maximal with respect to cardinality and replace it with inclusion maximality. Then, we optimize on the cardinality of X. This allows our proposed algorithms for this local problem to be executable in polynomial time with respect to the size of the data.

We propose a solution for the Global Induced Bipartite Subgraph Problem by reducing it to a corresponding Boolean satisfiability (SAT) problem which can subsequently be solved using a high-performance SAT-solver. On the other hand, we propose three polynomial-time heuristics for the Local Induced

Bipartite Subgraph Problem with different runtime-performance trade-offs. Then, we evaluate all four algorithms against each other on common benchmark graph datasets and compare their runtime and the quality of the results, i.e., the cardinality of the computed induced bipartite subgraphs. Our results show that for the local problem, the greedy heuristic has the best runtime while the genetic algorithm performs best with respect to the order of the computed bipartite subgraph. Finally, the employed simulated annealing algorithm balances runtime and performance.

We investigate this task because of two applications in this thesis where the discovery of such subgraphs is a necessity. In Chapter 5, we use them for the discovery of ordinal two-factorizations while in Chapter 7, we leverage them for an application in order diagram drawing. There are other possible fields where the application of this method is thinkable, for example cleaning datasets that are by their nature bipartite but are extracted from non-bipartite relations and thus have noise that makes them non-bipartite. An example for this could be author-to-publication networks. Also in the realm of system biology and medicine, bipartite graphs are of interest [95] and these areas could thus profit from our work. Furthermore, the approach could be used to discover hidden two-mode networks in graph data.

3.2 Related Work

Bipartiteness of graphs is a so-called *hereditary property*, i.e., every subgraph of a bipartite graph is once again bipartite. Lewis and Yannakakis [85] show in their work from 1980 that for any hereditary property $\mathcal{P}$, it is NP-complete to decide, whether a graph G can be made to satisfy $\mathcal{P}$ by deleting k vertices. Even approximations are known to be in the same complexity class [88].

Cohen et al. [19] provide an algorithm to compute all maximal induced subgraphs with a desired hereditary property and Trukhanov et al. [108] give a framework on how to design an algorithm to find a maximum vertex subset [108] with given hereditary property. Both approaches result in exponential algorithms.

A closely related problem is the edge-deletion problem. Here, instead of vertices, a set of edges is requested that makes the graph bipartite if it is deleted. Similarly to the vertex-deletion problem, deciding on the number of edges that have to be deleted is NP-complete [118].

Another well-investigated problem about subgraphs with a hereditary property is the clique-problem. For a given graph, the largest fully-connected subgraph is sought-after. Its NP-completeness was first proven by Karp in 1972 [71]. To tackle this problem, there are exact algorithms [14] as well as heuristics [116].

The Boolean satisfiability problem is the problem to determine for a given Boolean formula, i.e., a formula built from variables, conjunctions, disjunctions, and negations, whether there is a valid assignment of the variables such that the formula evaluates to TRUE. Each Boolean formula can be reformulated to its conjunctive normal form (CNF), which is considered to be the canonical way to represent it. A CNF consists of a conjunction of multiple clauses, which are in turn disjunctions of variables and negated variables. Deciding if an instance of the Boolean satisfiability problem is solvable is an NP-complete [21] task. So-called SAT-solvers, such as MiniSat [44], investigate instances of this problem and, if the instance is solvable, try to provide valid variable assignments in reasonable time. In this work, we make use of SAT-solvers by deducing the largest induced bipartite subgraph of a graph from a valid assignment of a Boolean formula.

Furthermore, we propose three heuristics that compute locally maximal subgraphs: a greedy algorithm, a simulated annealing algorithm and a genetic algorithm. Simulated annealing and genetic algorithms are well-established tools in algorithm engineering. We refer the reader to surveys [105, 72] to get an overview on the variations of both approaches.

3.3 Dual Formulation of the Problems

Even though we are interested in bipartite subgraphs of maximal order, we use an equivalent dual formulation from here on. For a graph $G = (V, E)$, we want to find a minimal set of vertices D such that the graph $G[V \setminus D]$ is bipartite. This is, as D can be interpreted as erroneous data that makes the graph non-bipartite.

Problem 3.3 (Global Vertex Deletion Problem)
Given: A graph $G = (V, E)$.
Requested: A subset of vertices $D \subseteq V$ such that the graph $G[V \setminus D]$ is bipartite and the cardinality of D is minimal.

By construction, a solution of the vertex deletion problem also provides a solution for the induced bipartite subgraph problem. This is, as for a graph $G = (V, E)$ with a minimal set D such that $G[V \setminus D]$ is bipartite, the set of $V \setminus D$ is a maximal set that induces a bipartite subgraph. Thus, maximizing the induced subgraph and minimizing the set of vertices to delete is equivalent. Similarly to the global problem, we also consider the dual formulation of the local problem.

Problem 3.4 (Local Vertex Deletion Problem)
Given: A graph $G = (V,E)$.
Requested: A set of vertices $D \subseteq V$ such that the graph $G[V \setminus D]$ is bipartite and there is no set $\tilde{D} \subsetneq D$ with the same property.
Optimization: Minimize $|D|$.

Once again an inclusion-maximal set $V \setminus D$ that induces a bipartite subgraph corresponds to an inclusion-minimal set D that has to be deleted.

3.4 Globally Maximal Bipartite Subgraphs

Checking if a graph is bipartite can be done in polynomial time with respect to the order of the graph. To do so a breadth-first search is performed where the vertices are colored in alternating colors conditional on their distance to the starting vertex, compare to Algorithm 3.1. Either the graph is bipartite, and the two color classes result in a valid bipartition, or the algorithm assigns some vertex to both color classes and thus an odd cycle that makes the graph non-bipartite exists.

Algorithm 3.1 Check whether a Graph is Bipartite

Input: Graph $G = (V,E)$ with $V = [n]$ for some $n \in \mathbb{N}$
Output: Report whether G is bipartite

```
def is_bipartite($V$,$E$):
    c = array of length $n$ initialized with $-1$
    for $v$ in $V$:
        if $c[v] = -1$:
            $c[i]$ = 0
            queue = $[i]$
            while queue not empty:
                $u$ = queue.pop_front()
                for $v$ neighbor of $u$:
                    if $c[v] = -1$:
                        $c[v]$ = $1 - c[u]$
                        queue.append($v$)
                    else if $c[u] = c[v]$:
                        return false
    return true
```

A naive approach to find a bipartite subgraph could check for all subsets of vertices, whether their deletion makes the graph bipartite. However, even for small examples, i.e., graphs with few vertices, this is infeasible, as checking all subset of cardinality k of a graph of order n will result in $\binom{n}{k}$ tests. For a more sophisticated approach to the problem, we reduce it to an instance of the Boolean satisfiability problem which we can then solve with a SAT-Solver, in our case MiniSat [44] in version 2.2. To be exact, we want to know for a graph $G = (V,E)$ on n vertices and m edges whether the deletion of k vertices can make the graph bipartite. Solving is done by finding a partition of V into the three sets A, B, D, such that A and B are independent sets and $|D| \leq k$.

Definition 3.5 (CNF Formulation of the Global Problem)
Let $G = (V,E)$ be a graph on n vertices and $k \in \mathbb{N}$. For each $v_i \in V$, define the three variables $V_{i,1}, V_{i,2}, V_{i,3}$ with the clauses

(i) $V_{i,1} \vee V_{i,2} \vee V_{i,3}$ for all $v_i \in V$ and
(ii) $\neg V_{i,1} \vee \neg V_{j,1}$ and $\neg V_{i,2} \vee \neg V_{j,2}$ for all $\{v_i, v_j\} \in E$.

Furthermore, add variables and clauses such the set $\{V_{i,3} \mid i \in [n]\}$ satisfies an at-most-k condition.

For the *at-most-k* condition, we make use of the variables and clauses introduced by Sinz [102]. Altogether, our SAT instance has $(n-1)(k+3)+3$ variables and $2m + 2nk + 2n - 3k - 1$ clauses. For this CNF, the following holds.

Proposition 3.6
The CNF from Definition 3.5 of a graph $G = (V,E)$ with $k \in \mathbb{N}$ has a valid assignment, if and only if the graph can be made bipartite by deleting k vertices. Then, for

$$\begin{aligned} A &:= \{v_i \mid V_{i,1} = TRUE\}, \\ B &:= \{v_i \mid V_{i,2} = TRUE \wedge V_{i,1} = FALSE\}, \\ D &:= \{v_i \mid V_{i,3} = TRUE\} \end{aligned}$$

the graph $G[V \setminus D]$ is bipartite with bipartition classes A and B.

Proof. Let the conjugative normal form have a valid assignment. Then, the set D has cardinality at most k as at most that many of $V_{i,3}$ can be true. Note, that every vertex that is not in D has to be in exactly one of A or B because of their definition and condition (i) of the conjugative normal form. Assume A and B are not bipartition classes of $G[V \setminus D]$. Then, there have to be two vertices in either A or B that are connected by an edge, without loss of generality in A. But this is a contradiction to the definition of A and condition (ii) of the conjugative normal form. Assume now on the other hand, that there is a set $\tilde{D}$ of cardinality at most k, such that $G[V \setminus D]$ is bipartite. Call the bipartition sets $\tilde{A}$ and $\tilde{B}$. But then $V_{i,1} = TRUE \iff v_i \in \tilde{A}$ and $V_{i,2} = TRUE \iff v_i \in \tilde{B}$ and $V_{i,3} = TRUE \iff v_i \in \tilde{D}$ is a valid assignment of the CNF. □

Now, we can build this SAT instance for each k increasing from 1 until we achieve a satisfiable instance. Then, the set $\{v_i \mid V_{i,3} = TRUE\}$ is exactly the subset of vertices we have to remove to make the graph bipartite. As a speed-up technique, we apply a binary search where at the beginning k is doubled in each step until an initial solution is found. Then, in each step the mean value of the known upper and lower bounds is checked until the exact solution is found.

3.5 Heuristics for the Local Problem

In this section, we now propose the three different heuristics to solve the local version of the problem, i.e., Problem 3.4.

3.5.1 Greedy

Our greedy algorithm, formalized in Algorithm 3.2, consists of the main routine `greedy` and the subroutine `greedy_fill`. The subroutine has to be called with the vertex set V partitioned into A_1, A_2, and D and neither A_1 nor A_2 is allowed to contain two vertices that are connected by an edge. The algorithm checks for all vertices u in D in a random order, whether they already have a neighbor in one of the bipartition classes A_1 or A_2. Then, the set N of the algorithm contains all indices $i \in \{1, 2\}$ such that u can be added to A_i without violating the previous assignments. If N is empty, u cannot be added to either bipartition class and therefore stays in D, otherwise, one of the possible classes in A_i with $i \in N$ is selected at random. Thus, the routine moves elements from D to A_1 and A_2 until all elements in D are connected with an edge to an element in A_1 and in A_2.

To compute an inclusion-minimal set D such that the induced graph on $V \setminus D$ is bipartite, we can now call this subroutine with V as D and empty sets for A_1 and A_2. The algorithm then computes D and the bipartition classes A_1 and A_2 of the resulting graph. By design, we can rely on this subroutine for the heuristics later proposed in this work.

Algorithm 3.2 Greedy Discovery of Induced Bipartite Subgraphs

Input: Graph $G = (V, E)$
Output: Inclusion-minimal set $D \subseteq V$, such that $G[V \setminus D]$ is bipartite
Bipartition classes A and B of $G[V \setminus D]$

```
def greedy($V$,$E$):
    return greedy_fill($V$,$E$,{},{},$V$)

def greedy_fill($V$,$E$,$A_1$,$A_2$,$D$):
    for $u$ in random_order($D$):
        $N = \{i \in \{1,2\} \mid \nexists v \in A_i, \{u,v\} \in E\}$
        if $N \neq \emptyset$:
            $D$.remove($u$)
            $i$ = random_element($N$)
            $A_i$.add($u$)
    return $A_1$,$A_2$,$D$
```

3.5.2 Simulated Annealing

Simulated annealing algorithms are motivated by the physical process of annealing metal which involves controlled cooling in order to achieve better physical properties. Classical hill-climbing algorithms generate a starting solution which is then improved iteratively by choosing neighboring solutions of better quality. By design, these algorithms often terminate in a local minimum. The simulated annealing approach tries to overcome this issue by accepting to worsen the current solution using a probability function which allows the algorithm to leave local minima. At the beginning, the algorithm accepts worse solutions with a high probability which is then iteratively decreased towards zero based on a cooling function. Thus, in the last iterations, the algorithm behaves similar to a hill-climbing algorithm.

Our version of the simulated approach in Algorithm 3.3 is initialized with a maximal number of iterations $i_{\max}$, a starting temperature $t_{\max}$ and a cooling function which maps to a range from 0 to $t_{\max}$. The initial solution is generated using the greedy algorithm from the previous section. In each iteration, a neighboring solution is chosen by selecting a random vertex u from D and removing all vertices that are connected by an edge to u from the bipartition classes A and B. Then, u can be added to A or B, from which one is chosen at random. Finally, the `greedy_fill` routine is used to ensure that the set D is minimal. Note, that using a cooling function that maps every value to zero results in a hill-climbing algorithm.

Algorithm 3.3 Simulated Annealing Discovery of Induced Bipartite Subgraphs

Input: Graph $G = (V, E)$
Maximal number of iterations $i_{\max}$
Starting temperature $t_{\max}$
Cooling function $cooling(i_{\max}, t_{\max}, i) \mapsto \mathbb{R}_{\geq 0}$
Output: Inclusion-minimal set $D \subseteq V$, such that $G[D]$ is bipartite
Bipartition classes A and B of $G[D]$

```
def simulated_annealing($V,E,i_{\max}$):
    $A,B,D$ = greedy($V,E$)
    for $i$ in [$i_{\max}$]:
        $A_c,B_c,D_c$ = compute_neighbor($V,E,A,B,D$)
        $c$ = $|D| - |D_c|$
        if $c > 0$:
            $A,B,D$ = $A_c,B_c,D_c$
        else:
            $t$ = cooling($i_{\max},t_{\max},i$)
            if $e^{c/t}$ > random(0,1):
                $A,B,D$ = $A_c,B_c,D_c$
    return $A,B,D$

def compute_neighbor($V,E,A,B,D$):
    $u$ = random_element($D$)
    for $v$ in neighbors($u$):
        $D$.add($v$)
    $D$.remove($u$)
    random_element($\{A,B\}$).add($u$)
    return greedy_fill($V,E,(A \setminus D),(B \setminus D),D$)
```

3.5.3 Genetic Algorithm

Finally, we propose an adaptation of a genetic algorithm which is motivated by the evolutionary process of reproduction. For this, a set of starting individuals is chosen. The ones with the highest fitness with respect to the optimization task are allowed to reproduce to achieve a new generation of individuals. Furthermore, mutations can be introduced to avoid local minima. This process is repeated until a good solution is found. In our version of this approach, which is formalized in Algorithm 3.4, we provide a maximal number of generations $g_{\max}$, a number of individuals $i_{\max}$ that exist in each generation and a mutation probability p_{mut}. At the beginning, we initialize $i_{\max}$ starting individuals using the greedy algorithm from Section 3.5.1. In each iteration of the loop, a new generation *next_gen* of individuals is generated which replaces the current set of individuals *cur_gen*

Algorithm 3.4 Genetic Discovery of Induced Bipartite Subgraphs

Input: Graph $G = (V, E)$
Maximal number $i_{\max}$ of individuals
Probability p_{mut} for mutations
Maximal number $g_{\max}$ of generation
Output: Inclusion-minimal set $D \subseteq V$, such that $G[D]$ is bipartite
Bipartition classes A and B of $G[D]$

```
def genetic($V$,$E$,$g_{\max}$,$i_{\max}$):
    $cur\_gen$ = []
    for $i$ in [$i_{\max}$]:
        $cur\_gen$.add(greedy($V$,$E$))
    for $i$ in [$g_{\max}$]:
        $next\_gen$ = []
        $P$ = generate_probability_distribution($cur\_gen$)
        for $j$ in [$i_{\max}$]:
            $(A_1,B_1,D_1),(A_2,B_2,D_2)$ = choose_with_prob($cur\_gen,P,2$)
            $next\_gen$.add(breed($V,E,A_1,B_1,D_1,D_2,p_{\text{mut}}$))
        $cur\_gen$ = $next\_gen$
    return (A,B,D) from $cur\_gen$ with $|D|$ minimal

def breed($V,E,A,B,D_1,D_2,p_{\text{mut}}$):
    $D_n$ = $D_1 \cup D_2$
    if random[0,1) < $p_{\text{mut}}$:
        return compute_neighbor($V,E,(A \setminus D_n),(B \setminus D_n),D_n$)
    else:
        return greedy_fill($V,E,(A \setminus D_n),(B \setminus D_n),D_n$)
```

with a new one. For this, a probability distribution based on the fitness of the current generation is computed. We choose the probability distribution such that the probability is linear with respect to the cardinality of D and that the element with the smallest cardinality D has ten times the probability of the one with the largest cardinality D. The elements of *next_gen* are generated by choosing two distinct elements for each based on the probability distribution and breeding them to a new individual. This is done by using the `breed` function which first computes the set D_n, as the union of both D-sets from the chosen individuals. Because bipartiteness is a hereditary property and D_n is a superset of the D-sets, the graph $G[V \setminus D_n]$ is also bipartite. Finally, the new individual is generated using the routine `greedy_fill` to ensure that the resulting D is inclusion-minimal. Alternatively, with probability p_{mut}, a mutation is introduced using the neighborhood choosing from the simulated annealing algorithm.

3.6 Evaluation and Discussion

In this section, we discuss and compare the different approaches proposed in this work with respect to their runtime and the quality of their results.

3.6.1 Datasets and Implementation

We test our algorithm on four different datasets.

DAGmar. The DAGmar [6] generator can produce random leveled graphs for given vertex numbers and densities. We use the graphs bundled with the DAGmar generator as our first testing dataset. Those have between 20 and 400 vertices and an edge-vertex ratio between 1.6 and 10.6.

Rome and North. The *Rome* and *North* (AT&T) graphs [20] are two datasets that are well-known benchmark graphs and are commonly applied by the graph drawing community.

Random. This class contains randomly generated Erdős–Rényi graphs on vertices between 10 and 500 vertices. The edge-vertex ratio was chosen such that it is between 1 and 10.

All of our experiments are implemented in Python 3. For the SAT-solver, we used the binaries provided on the MiniSat website [44]. The experiments are conducted on an Intel Xeon Gold 5122 CPU equipped with 300 GB RAM. For reproducibility and further research, our source code [35] is public.

3.6.2 Parameter Tuning

Our simulated annealing approach and our genetic algorithm require parameters which we have to choose carefully. In this section, we provide some motivation and hints on how to choose the parameters.

Simulated Annealing. The simulated annealing algorithm has three parameters to optimize. For the starting temperature, we observe that the average cardinality of D for all graphs in the test sets decreases until a temperature of around 45. Thus, we recommend using a starting temperature of 50. To decide on the cooling function, we provide the average result of the tests on four different cooling functions in Table 3.1. The results suggest that the quadratic cooling function should be plugged. Finally, the number of iterations that algorithm runs for can

Table 3.1: Different cooling functions for the simulated annealing algorithm. The lowest average cardinality of D was achieved with the quadratic cooling function.

Name	$cooling(i_m, t_m, t)$	Average cardinality of D
Hill Climbing:	0	20.50
Linear:	$t_m\left(\frac{i_m-t}{i_m}\right)$	19.87
Quadratic:	$t_m\left(\frac{i_m-t}{i_m}\right)^2$	19.39
Exponential:	$t_m\left(\frac{1}{1+e^{\frac{2\ln(t_m)}{i_m}\left(t-\frac{i_m}{2}\right)}}\right)$	19.91

be optimized. Our observations suggest that the improvements after more than 10 000 iterations are marginal, and we thus recommend this parameter.

Genetic. The genetic algorithm has three parameters. As a general rule, we observe that increasing the number of individuals or the number of generations improves the result. We are not able to observe a maximum number of individuals or generations beyond which the cardinality of D does not become smaller. We thus cap our algorithm at 20 individuals and 1000 generations. For the mutation probability, Figure 3.1 suggests that a mutation rate of 0.8 provides the best results. It is noteworthy, that the different spikes of Figure 3.1 stem from different optima from the different datasets. Thus, performing parameter optimization on each dataset is advisable when the algorithm is applied. We also experimented with survivors between generations however, we observed that this increases the cardinality of D.

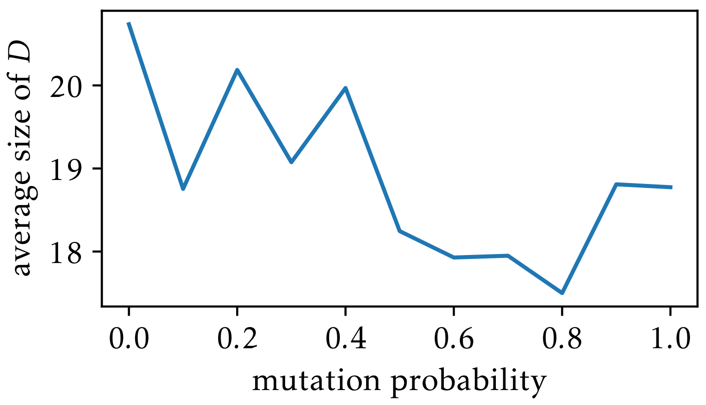


Figure 3.1: Average results for different mutation probabilities the genetic algorithm. The lowest average result for the cardinality of D was achieved with a mutation probability of 0.8.

3.6.3 Evaluation

We first consider the theoretical runtime of the different algorithms. Even though the transformation to the SAT instance is polynomial, the SAT-solver has by its nature an exponential runtime. The greedy algorithm has to process every edge once and is linear in the number of edges of the graph. In the worst case, the simulated annealing algorithm does a full procession of the greedy algorithm in each repetition and has therefore a runtime of $\mathcal{O}(m \cdot i_{\max})$. Finally, the runtime of the genetic algorithm is in $\mathcal{O}(m \cdot g_{\max} \cdot i_{\max})$, as in every repetition of the loop the greedy algorithm is called for each individual.

For the experiment-based evaluation, we choose our parameters as described in the last section. We fist consider the experimental runtime. Each graph is processed in our test-datasets with each algorithm once. If the SAT-algorithm runs longer than one hour, we cancel its computation. For the DAGmar graph class, only the computation of 304 from 1960 graphs finishes in this time, for the random class, it is 89 of 403. In Tables 3.2 and 3.3, we compare the average runtime and cardinality D of the different algorithms split by graph class. If we compare the heuristics, it is as expected by the theoretical runtime investigation: The greedy algorithm is the fastest one and the genetic algorithm is the slowest with the simulated annealing algorithm being in the middle. When we consider the cardinality of D, it is the other way round: The best results are achieved by the genetic algorithm and the worst ones by the greedy algorithm on all test datasets. The SAT-algorithm is, by its nature of being an exact algorithm, always the one with the smallest set D. A more visual representation of the time requirement is provided in the plot of Figure 3.3. Here, each graph in every class is represented

Table 3.2: Average cardinality of D with standard deviation of the four algorithms on all graph classes.

Cardinality of D	DAGmar	North	Random	Rome
SAT (avg)	18.12*	2.02	14.65*	3.93
SAT (sd)	10.07*	2.89	10.39*	2.36
Greedy (avg)	114.37	3.04	127.44	5.84
Greedy (sd)	72.34	4.41	91.11	3.81
Simulated Annealing (avg)	97.81	2.05	108.75	4.16
Simulated Annealing (sd)	64.43	2.94	80.13	2.52
Genetic (avg)	93.32	2.02	103.44	3.95
Genetic (sd)	61.91	2.89	76.81	2.38

*This algorithm did not finish in under one hour for all graphs of this class.

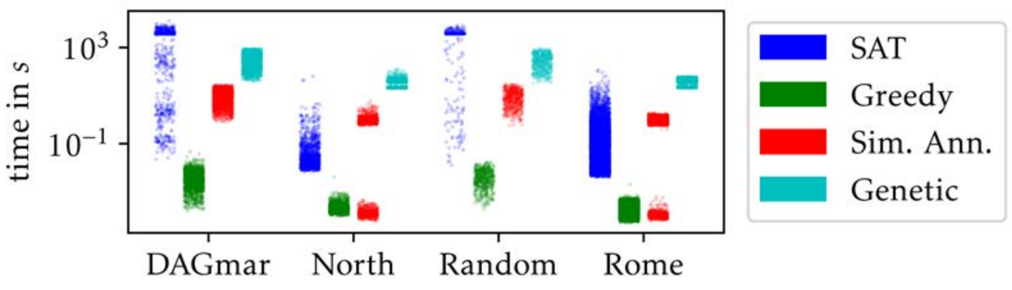


Figure 3.2: Time required by the different algorithms. Each dot represents a graph in the given class.

by a dot which is placed on a logarithmic scale in the y-axis, depending on the time, its computation needs for each algorithm. In Figure 3.3, we plot for all graph classes and algorithms how many graphs have a computed set D that is larger than k. To provide comparability, the graphs that did not finish in the SAT-experiment are excluded from this plot. Once again, we can observe that the greedy algorithm performs worst while the simulated annealing and genetic algorithm come very close to the SAT-algorithm which, by its nature of being an exact algorithm, is always the lowest curve in all four plots.

3.6.4 Discussion

Each of the four algorithms proposed in this work has its merits, as they balance runtime against cardinality of D in differently. An exact result is usually preferred, however, because of the exponential nature of SAT-solving, we observe that this approach is not guaranteed to finish in under an hour even for medium-sized graphs. On the other hand, the greedy algorithm has a runtime which is linear in

Table 3.3: Average time consumption and standard deviation of the four algorithms on all graph classes.

Time	DAGmar	North	Random	Rome
SAT (avg)	273.93s*	0.23s	420.51s*	0.46s
SAT (sd)	858.47s*	2.49s	1055.62s*	2.41s
Greedy (avg)	4.51ms	0.26ms	4.77ms	0.25ms
Greedy (sd)	3.67ms	0.17ms	3.72ms	0.12ms
Simulated Annealing (avg)	9.39s	0.66s	9.54s	0.98s
Simulated Annealing (sd)	6.16s	0.68s	6.75s	0.3s
Genetic (avg)	324.43s	35.88s	306.97s	43.51s
Genetic (sd)	217.8s	15.41s	210.01s	7.16s

*This algorithm did not finish in under one hour for all graphs of this class

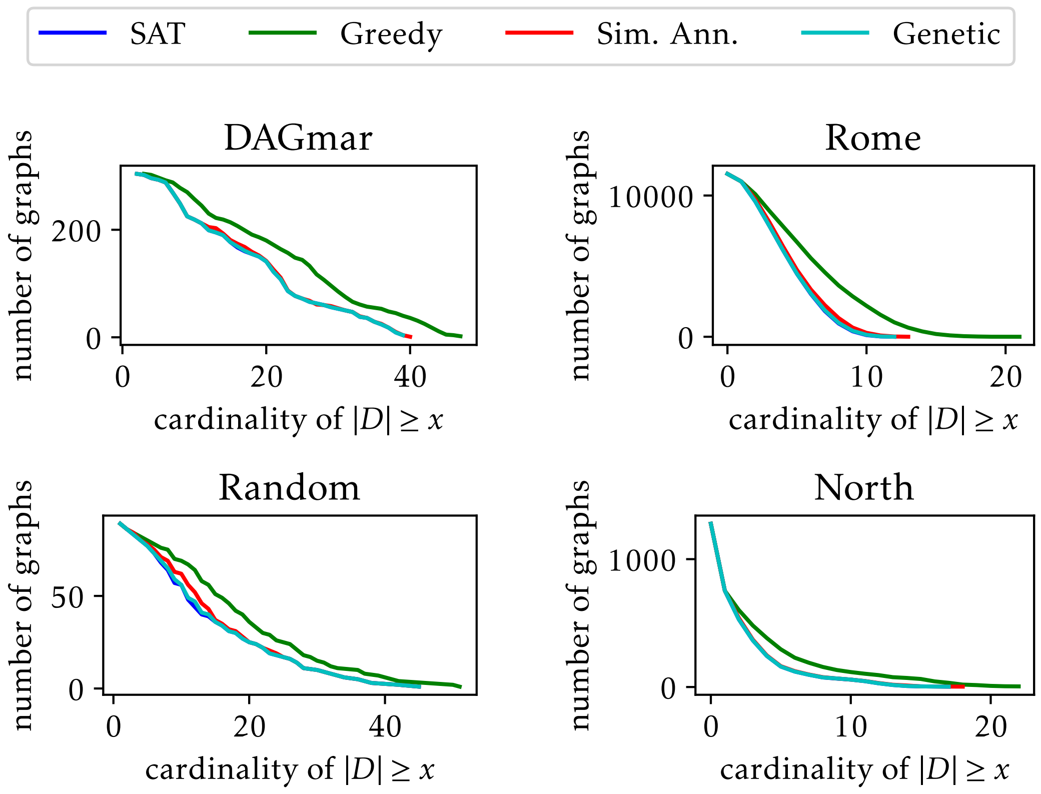


Figure 3.3: Number of graphs that where the cardinality of the computed D is at least x for the different algorithms. The lines of the SAT-algorithm, the simulated annealing algorithm, and the genetic algorithm are hard to distinguish from each other, as they performed very closely.

the number of edges of the graph and can thus usually be computed even on large graphs. The order of the computed bipartite subgraphs are not too unreasonable when compared to the exact solution of the SAT-algorithm. Finally, the simulated annealing and greedy algorithm are in the middle of the other two approaches in both regards, runtime and quality. The simulated annealing approach is a bit faster but delivers worse results. However, it should be noted that both of them are very close to the exact solutions. For a general recommendation, one should usually make use of the algorithm with the highest runtime one can afford.

3.7 Conclusion

In this section, we proposed one exact algorithm to compute maximal induced bipartite subgraphs and three heuristics to compute locally maximal induced bipartite subgraphs of large order. To this end, we employed a SAT-solver for the exact solution of the global problem. For the local problem, we designed three different heuristic approaches with a greedy strategy, a simulated annealing approach, and a genetic algorithm. Furthermore, we compared the results on four benchmark datasets and demonstrated in an experimental evaluation that all three heuristics have a reason to exist by balancing the time-consumption and the order of the computed subgraph to different degrees.

A notable observation is that the simulated annealing approach and the genetic heuristics could, by their design, work on any hereditary graph property. For this, only the greedy heuristic and the neighbor-choosing have to be adjusted. This raises the question, whether the approach generalizes to other hereditary graph properties. To confirm that these approaches work on other classes, further studies are necessary and are a natural future extension to this work. As we published our code, such experiments can easily be conducted.

CHAPTER 4

Contranominal Scales in Formal Contexts

One of the main goals of formal concept analysis is to enable humans to comprehend information encapsulated in the bipartite data. The main tool to enable this is the concept lattice. Thus, the number of concepts is a limiting factor for a human to understand underlying structural properties. The size of the concept lattice is influenced by the number of subcontexts in the corresponding formal context that are isomorphic to a contranominal scale of high dimension. This motivates the research question that we address in this chapter.

(1.2) How can we discover contranominal scales in bipartite graphs?

For this purpose, we propose the algorithm ContraFinder which enables the computation of all induced contranominal scales of a formal context. We evaluate our algorithm against two baseline algorithms to prove its merit. Our novel algorithm enables strategies such as δ-adjusting that decrease the number of contranominal scales in a formal context by the selection of an appropriate attribute subset.

4.1 Introduction

One of the main objectives of formal concept analysis is to enable humans to understand complex data. For this, the data is clustered into concepts which are then ordered in a lattice structure. Relationships between the features are represented as implications. However, the size of the corresponding concept lattice can increase exponentially in the size of the input data. As humans tend to comprehend connections in smaller chunks, the understandability is decreased by this exponential nature, even in medium-sized datasets. One way to optimize the readability of the concept lattices are nested line diagrams [112]. Another way are better drawing algorithms, as proposed in Chapters 6 and 7, to optimize the presentation of the concept hierarchy. However, both of them reach their limits, once the lattices grow too large. It is of interest to investigate why the number of concepts of even medium-sized datasets grows that large in the first place.

In an attribute combination of size k forms together with some of its objects a contranominal scale, these attributes will generate 2^k concepts in the concept lattice. The emerging structure is called a *k-dimensional Boolean suborder*. On the other hand, a lattice contains such a Boolean suborder if and only if the corresponding formal context contains an induced k-dimensional contranominal scale [4, 79]. Therefore, the number of contranominal scales which appear as a subcontext in a formal context is heavily related to the size of its concept lattices. Because of the exponential relation between the number of concepts in a Boolean suborder and its number of attributes, the following problem is of interest.

Problem 4.1 (Contranominal Problem)
Given: Let $\mathbb{K}$ be a formal context and k a natural number.
Requested: Is there an induced contranominal scale of dimension k in $\mathbb{K}$?

The goal of this chapter is to develop an efficient algorithm for finding all attribute combinations that contain contranominal scales and extracting them. Since deciding the Contranominal Problem for a given $\mathbb{K}$ and $k \in \mathbb{N}$ is NP-complete, we do cannot hope to find polynomial-time algorithms. We propose several algorithms in this chapter which combined compose ContraFinder. Our experiments show that ContraFinder is more effective at finding induced contranominal scales in real-world datasets than previous approaches.

Algorithms such as the one proposed in this chapter, which allow for the computation of all contranominal scales, are a necessity for approaches such as δ-

`adjusting`, which do attribute selection based on the number of contranominal scales [34] each attribute is contained in. In this, the number of attribute combinations that give rise to contranominal scales are required and weighted.

4.2 Related Work

Besides the number of concepts, which strongly correspond to the contranominal scales in the formal context $\mathbb{K}$ [3], some other measures can be used to evaluate the size and complexity of a concept lattice $\underline{\mathfrak{B}}(\mathbb{K})$. One of those is the *2-dimension* [106], which is the minimal dimension of a Boolean lattice into which $\underline{\mathfrak{B}}(\mathbb{K})$ is order-embeddable. On the other hand, the *breadth* [107] of a (concept) lattice $\underline{\mathfrak{B}}(\mathbb{K})$ refers to the maximum dimension of a Boolean lattice that can be order-embedded into $\underline{\mathfrak{B}}(\mathbb{K})$. A connection between the concept lattices of a context $\mathbb{K} = (G, M, I)$ and its subcontext $\mathbb{S} = \mathbb{K}[H, N]$ is given by Ganter and Wille [57, Prop. 32] through the order embedding

$$\varphi : \underline{\mathfrak{B}}(\mathbb{S}) \to \underline{\mathfrak{B}}(\mathbb{K}),\ (A, B) \mapsto (A, A').$$

If $G = H$, this map is meet-preserving. Albano and Chornomaz [4] use this map to connect induced contranominal scales to Boolean suborders.

Proposition 4.2 (Albano and Chornomaz [4])
Let $\mathbb{K}$ be a formal context. If $\mathbb{K}$ contains a contranominal scale of dimension k then $\underline{\mathfrak{B}}(\mathbb{K})$ has a Boolean suborder of dimension k.

The other direction of this is demonstrated by Koyda and Stumme [79].

Proposition 4.3 (Koyda and Stumme [79])
Let $\mathbb{K}$ be a formal context. For every Boolean suborder $\underline{S}$ in the concept lattice of dimension k, there is an induced contranominal scale of the same dimension in $\mathbb{K}$.

Hence, a lattice contains an n-dimensional Boolean suborder if and only if its formal context contains an n-dimensional contranominal scale. These statements provide the connection between large induced contranominal scales and their relationship to the number of concepts.

4.2.1 Contranominal Scales and Induced Matchings

The complexity of computing contranominal scales in formal contexts is well investigated, as it can be considered to be the dual problem of finding induced maximum matchings in bipartite graphs. A matching is a subset of the edges, such that no two edges share a vertex. Therefore, an induced matching is a subset of the vertices that induce a matching on their edges. The decision problem underlying the computation of induced matchings is formulated as follows.

Problem 4.4 (INDUCED MATCHING PROBLEM)
Given: Let G be a bipartite graph and k a natural number.
Requested: Is there an induced matching of size k in G?

The complexity of this problem is well-investigated.

Theorem 4.5 (Lozin [87])
Deciding the INDUCED MATCHING PROBLEM is NP*-complete.*

The relationship between the INDUCED MATCHING PROBLEM and the CONTRANOMINAL PROBLEM follows directly from their respective definitions.

Lemma 4.6 (Contranominal-Matching-Duality)
Let (S,T,E) be a bipartite graph, $\mathbb{K} := (S,T,(S\times T)\setminus E)$ a formal context such that $H \subseteq S, N \subseteq T$. The edges between H and N compose an induced matching of size k in (S,T,E) if and only if $\mathbb{K}[H,N]$ is an induced contranominal scale of dimension k.

Proof. The complement context of (G,M,I) corresponds to the bipartite graph $(G,M,(G\times M)\setminus I)$. Thus, the statement follows by the definitions of induced matching and contranominal scale. □

This duality then directly gives rise to the following.

Theorem 4.7 (NP-completness of the CONTRANOMINAL PROBLEM)
Deciding the CONTRANOMINAL PROBLEM is NP*-complete.*

Proof. Follows directly from the duality of the CONTRANOMINAL PROBLEM and the INDUCED MATCHING PROBLEM. □

Thus, provided that $\mathsf{P} \neq \mathsf{NP}$, we can not expect to find polynomial-time algorithms for computing maximal bipartite subgraphs.

4.2.2 Baseline Algorithms

We compare our approach to two benchmark algorithms. The first one is an algorithm to compute induced matchings in bipartite graphs, the second one is an algorithm that computes cliques. We can apply the clique algorithm because of a relation of clique and contranominal scales that we will discuss in this section.

4.2.3 Using Induced Matchings

One of our baseline algorithms is the branch and search approach by Xiao and Tan [117]. It exploits the fact that for each maximum matching and each vertex, there is either an adjacent edge to this vertex in the matching or each of its neighboring vertices has an adjacent edge in the matching. Branching on vertices decreases the order of the processed graphs iteratively. Note that this idea, in contrast to our approach ContraFinder, does not exploit the bipartiteness of the graph.

4.2.4 Using Cliques

The problem of computing contranominal scales is closely related to the problem of computing cliques in graphs, which is defined as follows.

Problem 4.8 (Clique Problem)
Given: Let G be a graph and k a natural number.
Requested: Is there a clique of order k in G?

The computational cost of the Clique Problem is also well-examined.

Theorem 4.9 (Karp [71])
Deciding the Clique Problem is NP-*complete.*

To investigate the connection between the Clique Problem and the Contranominal Problem, define the conflict graph as follows:

Definition 4.10 (Conflict Graph)
Let $\mathbb{K} := (G, M, I)$ be a formal context. Define the *conflict graph* of $\mathbb{K}$ as the graph $\mathrm{cg}(\mathbb{K}) := (V, E)$ with the vertex set $V = (G \times M) \setminus I$ and the edge set

$$E = \{\{(g,m),(h,n)\} \in \binom{V}{2} \mid (g,n) \in I, (h,m) \in I\}.$$

The relationship between the cliques in the conflict graph and the contranominal scales in the formal context is given through the following lemma.

Lemma 4.11 (Contranominal Scale and Clique Connection)
Let $\mathbb{K} = (G, M, I)$ be a formal context, cg($\mathbb{K}$) *its conflict graph and $H \subseteq G, N \subseteq M$. Then $\mathbb{K}[H, N]$ is a contranominal scale of dimension k if and only if $(H \times N) \backslash I$ is a clique of order k in* cg($\mathbb{K}$).

Proof. *"⇒".* Let $\mathbb{K} = (G, M, I)$ be a formal context and $\mathbb{S} = \mathbb{K}[H, N]$ a contranominal scale of dimension k such that $H = \{h_1, h_2, \ldots, h_k\}$, $N = \{n_1, n_2, \ldots, n_k\}$ and $(h_i, n_j) \in I$ if and only if $i \neq j$. As $(h_i, n_i) \notin I$ for all $i \in \{1, 2, \ldots, k\}$, the graph cg($\mathbb{K}$) contains all elements (h_i, n_i) as vertices. Assume two such vertices, without loss of generality (h_1, n_1) and (h_2, n_2), are not connected by an edge in cg($\mathbb{K}$). Then either $(h_1, n_2) \notin I$ or $(h_2, n_1) \notin I$, a contradiction to $\mathbb{S}$ being contranominal.

"⇐". Let $\{(h_1, n_1), (h_2, n_2), \ldots, (h_k, n_k)\}$ be the vertex set of the clique of order k in cg($\mathbb{K}$). Then $(h_i, n_i) \notin I$ and $h_i, n_j \in I$ for $i \neq j$ by definition of the conflict graph. Thus, $\mathbb{S} := (\{h_1, h_2, \ldots, h_k\}, \{n_1, n_2, \ldots, n_k\}, \{(h_i, n_j) \mid i \neq j\}) \leq \mathbb{K}$ and $\mathbb{S}$ is a contranominal scale. □

Building on Lemma 4.11, the set of all induced contranominal scales can be computed using algorithms that iterate all cliques in the conflict graph. The set of all cliques then corresponds to the set of all contranominal scales in the formal context. An algorithm to enumerate all cliques in a graph is proposed by Bron and Kerbosch [14].

4.3 ContraFinder

In this section, we introduce the recursive backtracking algorithm ContraFinder to compute all contranominal scales. As it has exponential runtime, we also propose two speedup techniques in the subsequent section.

The main idea behind ContraFinder is the following. In each recursion step a set of pairs corresponding to an attribute set is investigated:

Definition 4.12 (Characterizing Pair and Generator)
Let $\mathbb{K} = (G, M, I)$ be a formal context and $N \subseteq M$. Define

$$C(N) := \{(g, m) \notin I \mid g \in G, m \in N \text{ and } \forall x \in N \setminus \{m\} : (g, x) \in I\}$$

as the set of *characterizing pairs* of N. We call N the *generator* of $C(N)$.

The characterizing pairs encode all induced contranominal scales entailed in these attributes as follows.

Algorithm 4.5 ContraFinder for the Discovery of Contranominal Scales

Input: Formal Context $\mathbb{K} = (G, M, I)$
Output: Set of all Contranominal Scales

```
def compute_contranominal_scales($G,M,I$):
  characterizing_pairs($\emptyset,M,\emptyset,I$)

def characterizing_pairs($C_N,\tilde{M},F,I$):
  for $m$ in $\tilde{M}$ in lexicographical order:
    $\tilde{M} = \tilde{M} \setminus \{m\}$
    $cand\_C_N = \{(g,n) \in C_N \mid (g,m) \in I\}$
    $cand\_m = \{(g,m) \mid (g,m) \notin I, g \notin F, \nexists n : (g,n) \in C_N\}$
    if $|\{g \mid (g,n) \in C_N\}| = |\{g \mid (g,n) \in cand\_C_N\}|$ and $|cand\_m| > 0$:
      unpack_contranominals($cand\_C_N \cup cand\_m$)
      $C_{N_{\text{new}}} = cand\_C_N \cup cand\_m$
      $F_{\text{new}} = F \cup \{g \in G \mid (g,m) \notin I\}$
      characterizing_pairs($C_{N_{\text{new}}},\tilde{M},F_{\text{new}},I$)

def unpack_contranominals($C_N$):
  $N = \{m \mid (g,m) \in C_N\}$
  for O in $\{\{g_{m_1},\ldots,g_{m_{|N|}}\} \mid m_i \in N, g_{m_i} \in \{g \in G \mid (g,m_i) \in C_N\}\}$
    report $(O,N)$ as contranominal scale
```

Lemma 4.13
Let $\mathbb{K} = (G, M, I)$, $N \subseteq M$ and

$$H(m) := \{g \in G \mid (g,m) \in C(N)\}.$$

Then $\mathbb{K}[O,N]$ is a contranominal scale if and only if O contains exactly one element of each $H(m)$ with $m \in N$.

Proof. "$\Rightarrow$" Let $O = \{g_1, \ldots, g_{|N|}\}$ such that it contains exactly one element of each $H(m)$. Then, for every object $g \in O$, there is exactly one $m \in N$ with $(g,m) \notin I$ due to the definition of $C(N)$. Also, $|O| = |N|$ as $H(m) \cap H(n) = \emptyset$ for distinct $m,n \in M$. Thus, the context $\mathbb{K}[O,N]$ is a contranominal scale.

"$\Leftarrow$" Now let $\mathbb{S} = \mathbb{K}[O,N]$ be a contranominal scale. By definition for all elements $(h,n) \in C(N)$, it holds $(h,n) \notin I$. Because $\mathbb{S}$ is contranominal, there is no attribute $m \in N$ with two objects $g,h \in O$ such that $(g,m),(h,m) \notin I$. Thus, O contains exactly one element of each $H(m)$ with $m \in N$. □

Lemma 4.13 implies that such contranominal scales can exist only if no $H(m)$ is empty and $|N| = |O|$. Both sets can be reconstructed from a set of characterizing pairs corresponding to N. This is done in `unpack_contranominals` in Algorithm 4.5. Therefore, N does not have to be memorized in CONTRAFINDER. The algorithm exploits that for each set of characterizing pairs $C(N)$ the attributes N can be iterated in lexicographical order, similar to NEXTCLOSURE [55].

Definition 4.14 (Lexicographical Order)
Let $(M,\leq)$ be a linearly ordered set. The *lexicographical order* on $\mathcal{P}(M)$ is a linear order. Let $A = \{a_1,\ldots,a_n\}$ and $B = \{b_1,\ldots,b_m\}$ with $a_i < a_{i+1}$ and $b_i < b_{i+1}$. Then $A < B$ if $n < m$ and $(a_1,\ldots,a_n) = (b_1,\ldots,b_n)$ or if $\exists i : \forall j \leq i : a_j = b_j$ and $a_i < b_i$.

Similar to the TITANIC-algorithm, CONTRAFINDER utilizes the property that contranominal scales are anti-monotonic, i.e., that each contranominal scale of dimension k has a contranominal scale of dimension $k-1$ as subcontext. Thus, only attribute combinations N have to be considered if $\forall N' \subseteq N : C(N') \neq \emptyset$. The algorithm iterates using a set $\tilde{M}$ which starts with all attributes and removes in each recursion step an attribute in lexicographical order to guarantee that all attribute combinations of the formal context with contranominal scales are investigated. Furthermore, there is a set F of forbidden objects that increases in each recursion step. These are the objects that can't appear in a contranominal scale with the corresponding attributes, as they are non-incident to more than one attribute. This is possible, as each contranominal scale contains exactly one non-incidence in each contained object.

Theorem 4.15 (Correctness of CONTRAFINDER)
The algorithm CONTRAFINDER reports every contranominal scale exactly once.

Proof. We first show that the algorithm iterates over every contranominal scale of a formal context $\mathbb{K} = (G,M,I)$ at least once. Let $\mathbb{S} = \mathbb{K}[H,N]$ be a contranominal scale of dimension k not computed by the algorithm that does not contain a smaller contranominal scale that is not computed by the algorithm. Let $H = \{g_1,...,g_k\}$, $N = \{m_1,\ldots,m_k\}$ and $(g_i,m_j) \in I$ for all $i \neq j$. Without loss of generality $m_1 \leq m_2 \leq ... \leq m_k$ is the lexicographic order on N. Consider the contranominal scale $\tilde{\mathbb{S}} = \mathbb{K}[\{g_1,...,g_{k-1}\},\{m_1,\ldots,m_{k-1}\}]$ of dimension $k-1$ that is computed by the algorithm. Thereby, the generator of the characterizing pairs is given by $\{m_1,\ldots,m_{k-1}\}$. Thus, in the next iteration m_k is added to this generator and (g_k,m_k) is added to C_N. Due to the contranominal structure of $\mathbb{S}$, no element of H is contained in the forbidden set F and thus, no element of $\{(g_1,m_1),(g_2,m_2),...,(g_{k-1},m_{k-1})\}$ is eliminated from C_N. Therefore, C_N corresponds to the characterizing pairs in the next step of the algorithm and the contranominal scale $\mathbb{S}$ is reported.

We now show that the algorithm iterates over every contranominal scale at most once. As the algorithm iterates over the generator attribute sets in lexicographical order, no attribute combination is iterated twice and every contranominal scale is reported at most once. □

Note that the algorithm ContraFinder, combined with Lemma 4.6, can also be used to compute all maximum induced matchings in bipartite graphs.

4.4 Speedup Techniques for ContraFinder

We now propose two approaches that can improve the speed of finding contranominals by decreasing the size of the context.

4.4.1 Clarifying and Reducing

We now consider the clarified and reduced formal contexts and reconstruct the contranominal scales of the original context from the contranominal scales of the augmented one. This allows the processes clarifying and reducing, which are often performed in formal concept analysis, to be used to discover contranominals.

In the clarified context, each pair of objects or attributes is merged if equality of their derivations holds. To deduce the original formal context from the clarified one, the previously merged attributes and objects can be duplicated. Thus, contranominal scales containing merged objects or attributes are duplicated.

Now, we demonstrate how to reconstruct the contranominal scales from attribute reduced contexts. Thereby, for each eliminated attribute m, we have to memorize the irreducible attribute set that has the same derivation as m.

Definition 4.16
Let $\mathbb{K} = (G, M, I)$ be a formal context and $R(\mathbb{K})$ the set of all attributes that are reducible in $\mathbb{K}$. Define the map $\omega\colon R(\mathbb{K}) \to \mathcal{P}(M \setminus R(\mathbb{K}))$ with

$$x \mapsto (N \subseteq M \setminus (R(\mathbb{K}) \cup \{x\}))$$

such that $N' = x'$ and N of greatest cardinality. For a fixed object set $H \subseteq G$, let $\omega_H\colon R(\mathbb{K}) \to \mathcal{P}(M \setminus R(\mathbb{K}))$ be the map with

$$x \mapsto \{y \mid y \in \omega(x), \forall h \in H : (h,x) \notin I \Rightarrow (h,y) \notin I\}.$$

Note, that the map ω is well-defined, as the uniqueness follows directly from the maximality of N. The following lemma provides a way to reconstruct the contranominal scales in the original context from the ones in the reduced one.

Lemma 4.17
Let $\mathbb{K} = (G,M,I)$ be a formal context with $\mathbb{K}_r$ its attribute-reduced subcontext and $\mathcal{K}$ the set containing all contranominal scales of $\mathbb{K}_r$. Then the set

$$\tilde{\mathcal{K}} = \{\mathbb{K}[H,\tilde{N}] \mid \mathbb{K}[H,N = \{n_1,\ldots,n_l\}] \in \mathcal{K}, \tilde{N} = \{\tilde{n}_i \mid n_i = \tilde{n}_i \vee n_i \in \omega_H(\tilde{n}_i)\}\}$$

contains exactly all contranominal scales of $\mathbb{K}$.

Proof. Let $\mathbb{S} = \mathbb{K}[H,\tilde{N}]$ be a contranominal scale in $\mathbb{K}$. Then each attribute $\tilde{n}_i \notin R(\mathbb{K})$ and thus, an attribute of $\mathbb{K}_r$ or there is a unique minimal irreducible attribute set $U \subseteq M \setminus R(\mathbb{K})$ with $\tilde{n}_i' = U'$ due to the definition of reducibility. In particular, for all $g \in G$ with $(g,\tilde{n}_i) \in I$ it holds that $(g,u) \in I$ for all $u \in U$. Furthermore, for all $g \in G$ with $(g,\tilde{n}_i) \notin I$, there is at least one $u \in U$ with $(g,u) \notin I$. Due to the contranominal property of $\mathbb{S}$, for every $\tilde{n}_i \in N$, there is exactly one $h \in H$ with $(h,\tilde{n}_i) \notin I$. Therefore, there is at least one $n_i \in U$ with the same property and thus, $\mathbb{S} \in \tilde{\mathcal{K}}$. Now let $\mathbb{S} = \mathbb{K}[H,\tilde{N}]$ be an element of $\tilde{\mathcal{K}}$. For each pair $(g,m) \notin I$, there is no attribute n such that $(g,n) \notin I$ and there is no object h such that $(h,n) \notin I$. Due to the existence of a g_i for each m_i such that $(g_i,m_i) \notin I$, the context $\mathbb{S}$ is a contranominal scale. □

Thus, to reconstruct contranominal scales for each $x \in R(\mathbb{K})$, all $y \in \omega(x)$ are considered. $U \cup x$ is a candidate for the attribute set of a contranominal scale in $\mathbb{K}$, if there is a $U \subseteq M \setminus \omega(x)$ with $U \cup y$ attribute set of a contranominal scale $\mathbb{S}_y$ for all y. This candidate forms the contranominal scale $\mathbb{K}[H,U \cup x]$, if and only if all contranominal scales $\mathbb{S}_y$ share the same object set H. The object reducible case can be done dually.

4.4.2 Knowledge-Cores

The notion of (p,q)-cores is introduced to formal concept analysis by Hanika and Hirth in [63]. Thereby, dense subcontexts are defined as follows:

Definition 4.18 (Knowledge Cores, Hanika and Hirth [63])
Let $\mathbb{K} = (G,M,I)$ and $\mathbb{S} = \mathbb{K}[H,N]$ be formal contexts. $\mathbb{S}$ is called a (p,q)-*core* of $\mathbb{K}$ for $p,q \in \mathbb{N}$, if $\forall g \in H : |g'| \geq p$ and $\forall m \in N : |m'| \geq q$ and $\mathbb{S}$ is maximal under this condition whereby the derivations are the ones with respect to $\mathbb{K}$.

Every formal context with fixed p and q has a unique (p,q)-core. Computing knowledge cores provides a way to reduce the number of attributes and objects in a formal context without removing large contranominal scales.

Lemma 4.19
Let $\mathbb{K}$ be a formal context, $k \in \mathbb{N}$, and $\mathbb{S} \leq \mathbb{K}$ its $(k-1,k-1)$-core. Then for every contranominal scale $\mathbb{C} \leq \mathbb{K}$ of dimension k, it holds $\mathbb{C} \leq \mathbb{S}$.

Proof. Assume not; i.e., there is a contranominal $\mathbb{K}[H,N] \not\leq \mathbb{S} = \mathbb{K}[H_S,N_S]$. But then $\mathbb{K}[H_S \cup H, H_S \cup N]$ is a $(k-1,k-1)$-core of $\mathbb{K}$ and $\mathbb{S} \leq \mathbb{K}[H_S \cup H, H_S \cup N]$, contradicting the definition of $(k-1,k-1)$-cores. □

Thus, to compute all contranominal scales of dimension at least k, we can first compute the $(k-1,k-1)$-core. Note that, in this case, however, smaller contranominal scales might get eliminated. Therefore, if the goal is to compute contranominal scales of smaller dimensions, the $(k-1,k-1)$-cores should not be computed. This is in contrast to the clarifying and reducing approach, where we were able to reconstruct all contranominal scales.

4.5 Evaluation and Discussion

In this section, we evaluate the algorithm ContraFinder.

4.5.1 Datasets

Table 4.1 provides descriptive properties of the datasets used in this work. The *zoo* [122, 27] and *mushroom* [92, 27] datasets are classical examples often used in FCA based research such as the TITANIC algorithm. The Wikipedia [81] dataset depicts the edit relation between authors and articles while the *wiki44k* dataset is

Table 4.1: Datasets used for the evaluation of ContraFinder.

	Zoo	Students	Wikip.	Wiki44k	Mushroom
Objects:	101	1000	11273	45021	8124
Attributes:	43	32	102	101	119
Density:	0.40	0.28	0.015	0.045	0.19
Number of concepts:	4579	17603	14171	21923	238710
Mean obj. per concept:	18.48	16.73	20.06	109.47	91.89
Mean att. per concept:	7.32	5.97	5.88	7.013	16.69
Size of canonical base:	401	2826	4575	7040	2323

a dense part of the Wikidata knowledge graph. The original wiki44k dataset was taken from [65]. In this work, we conduct our experiments on an adapted version by [64]. Finally, the *students* dataset [101] depicts grades of students together with properties such as parental level of education.

4.5.2 Runtime

ContraFinder is a recursive backtracking algorithm that iterates over all attribute sets containing contranominal scales. Thus, the worst case runtime is given by $\mathcal{O}(n^k)$ where n is the number of attributes of the formal context and k the maximum dimension of a contranominal scale in it. The Branch-And-Search algorithm from [117] has a runtime of $\mathcal{O}(1.3752^n)$ where n is the sum of attributes and objects. Finally, the Bron-Kerbosch algorithm has a worst-case runtime of $\mathcal{O}(3^{n/3})$ with n being the number of non-incident object-attribute pairs.

To compare the practical runtime of the algorithms, we test them on the previously introduced real world datasets. We report the runtime in Table 4.2 together with the dimension of the larges contranominal scale and the total number of contranominal scales. Note that, for larger datasets, we are not able to compute the number of all contranominal scales using Bron-Kerbosch (from Students) and the Branch-And-Search algorithm (Mushroom) below 24 hours due to their exponential nature and thus stopped the computations. All experiments are conducted on an Intel Core i5-8250U processor with 16 GB of RAM.

4.5.3 Discussion

The theoretical runtime of ContraFinder is polynomial in the dimension of the maximum contranominal. Therefore, compared to the baseline algorithms, it performs better the smaller the maximum contranominal scale in a dataset.

Table 4.2: Experimental results comparing the runtime of the three algorithms that compute contranominal scales.

	Zoo	Students	Wikipedia	Wiki44k	Mushroom
ContraFinder:	**2.43s**	**7.36s**	**17.15s**	**35.65s**	**1961.0s**
Bron Kerbosch:	138.70s	>86400s	>86400s	>86400s	>86400s
Branch & Search:	14.40s	12005.82s	1532.17s	16783.58s	>86400s
Max. con. scale:	7	8	9	11	10
# con. scales:	$4.1 \cdot 10^7$	$7.8 \cdot 10^9$	$9.9 \cdot 10^8$	$2.0 \cdot 10^{14}$	$1.2 \cdot 10^{19}$

Furthermore, the runtime of Bron-Kerbosch is worse, the sparser a formal context, as the number of pairs that are non-incident increases and thus more vertices have to be iterated. Finally, the Branch-And-Search algorithm is best in the case that the dimension of the maximum contranominal scale is not bounded. To evaluate how this theoretical properties translate to real world data, we compute the set of all contranominal scales with the three algorithms on the previously described datasets. Only the ContraFinder algorithm was able to compute the set of all contranominal scales on the larger datasets on our hardware under 24 hours. We conclude, that the runtime of ContraFinder is thus superior to the two baselines on our tested real-world datasets.

4.6 Conclusion

Contranominal scales in formal contexts are a defining reason why the concept lattices of even small contexts can get large, while on the other hand, providing little to none structural information. It is thus an important research task to identify these contranominal scales. In this chapter, we proposed the algorithm ContraFinder to enable the computation of all contranominal scales of a formal context. The algorithm starts by computing all attribute combinations that are part of a contranominal scale. Then, the contranominal scales with these attribute combinations are computed. We also propose two speed-up techniques using previous notions of formal concept analysis by linking the clarification, reduction, and the computation of (p,k)-cores to the problem.

The computation of attribute combinations entailing contranominal scales, which is enabled by our work, allows approaches that measure the influence of attributes on the size of the concept lattice. To this end, there are approaches such as δ-`adjusting` which remove the attributes with large contranominal influence.

CHAPTER 5

Ordinal Factors in Formal Contexts

An ordinal factor is a subset of the incidence relation of a formal context that forms a chain in the concept lattice, i.e., a subset that corresponds to a linear order. These linear orders allow for the visualization and organization of information in the dataset. In small data, they can be used to enable a human to grasp connections encapsulated in the data. In larger datasets, ordinal factors are useful as a navigation tool. We thus approach the following research question.

(1.3) How can we discover ordinal factors in bipartite graphs?

To this end, we propose two algorithms. First, ORD2FACTOR allows computing factorizations into two large ordinal factors. Secondly, we propose the greedy algorithm ORDIFIND that leverages previous work of fast concept lattice computations. We present a way to use this new algorithm as a comprehensive tool to discover relationships in the dataset. We furthermore introduce a distance measure based on the representation emerging from the ordinal factorization to discover similar objects. To evaluate the method, we conduct a case study on different datasets. Both problems that we discuss in this section have underlying NP-complete decision problems.

5.1 Introduction

Providing systems that allow for the extraction of explanations and the discovery relationships in bipartite graphs is often a necessity. However, with growing datasets this rapidly becomes a challenging task. A common way to tackle this problem, is to treat the binary attributes as numerical, where the value 1 is assigned to an object if it has the attribute and 0 otherwise. Then, methods from the classical toolkit of dimensionality reduction, such as principal component analysis, are applied. These approaches merge the different tags into a few axes while weighting the attributes in each axis. An emerging axis thus yields information about the presence or absence of correlated features in the original dataset. Then, each object is assigned a real-valued number in each axis to represent whether this object has its attributes. As a single axis represents multiple merged attributes, the resulting placement of objects with only a part of attributes yields an ambiguous representation. Because of this, the main issues of this method arise. Assigning a real value to an object is not consistent with the level of measurement of the underlying binary data. The assigned value promotes the perception that an element has, compared to others, a stronger bond to some attributes, which is not possible in a bipartite graph. Thus, a method that encourages such comparisons and results in such an inaccurate representation of the original information is, in our opinion, not valid. An example of such a principal component analysis projection is given in Figure 5.1.

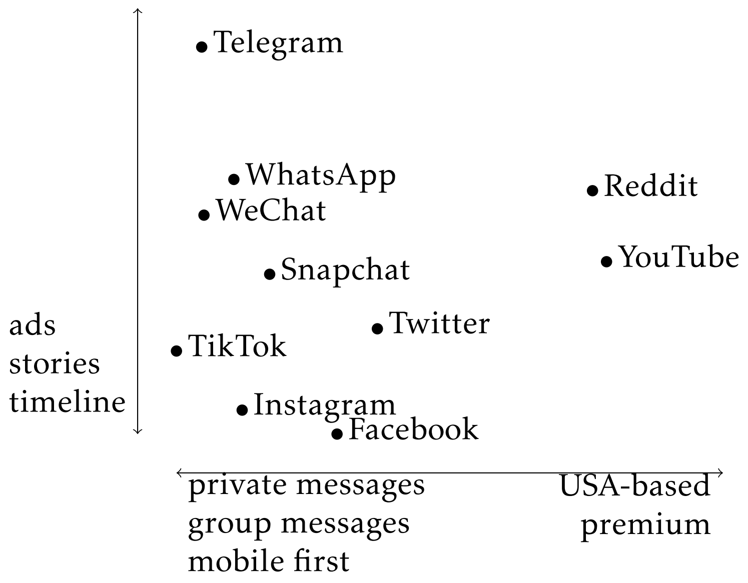


Figure 5.1: A 2-dimensional projection of the objects from the dataset depicted in Figure 5.3 using principal component analysis.

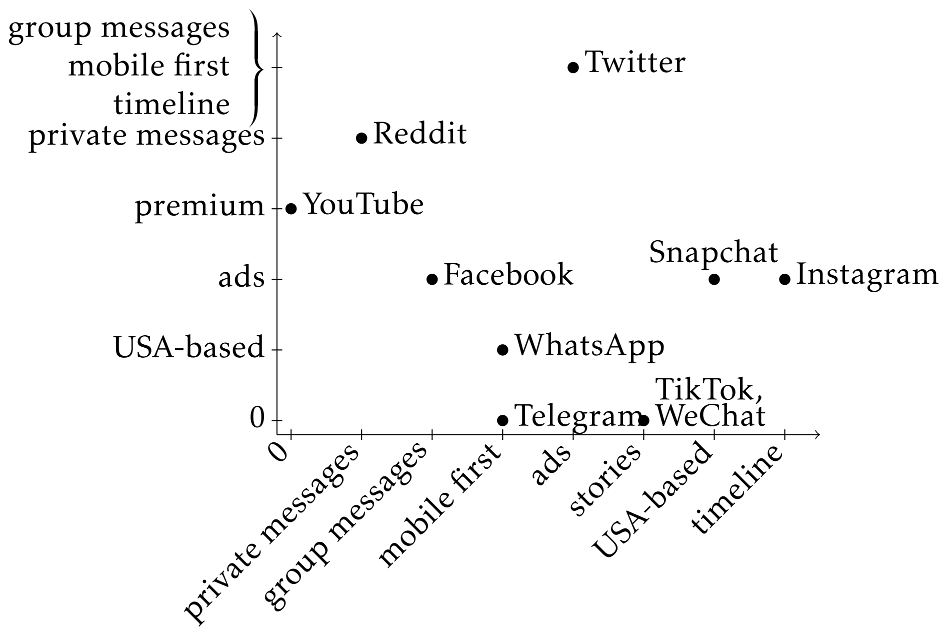


Figure 5.2: An ordinal factorization of the dataset depicted in Figure 5.3 restricted to the two largest factors for improved readability. All incidences can be deduced from the projection except: (TikTok, timeline), (Whatsapp, stories), (Facebook, timeline), (YouTube, stories), (Facebook, stories). The ordinal projection does not contain false data.

To address this problem, Ganter and Glodeanu [56] developed the *ordinal factor analysis*, a method that tackles this challenge without the use of real-valued measurement on the binary attributes. The main idea is to condense multiple attributes into a single factor, similar to principal component analysis. The projection consists of linear orders of attributes, the so-called *ordinal factors*. Then, the method assigns each object, based on its attributes, a position in every factor. Compared to the principal component analysis approach, the positions assigned in the process are natural numbers instead of real-valued ones. Thus, the resulting projection does not express inaccurate or even incorrect information. A complete ordinal factorization furthermore allows deducing all original information. Similar to the principal component analysis projection, there is a visualization method for small datasets. Thereby, one places each object in a two-dimensional coordinate system at the last position for each axis such that it has all attributes until this position. Such a projection can be seen in Figure 5.2. The main advantage of the ordinal factor plot compared to the principal component analysis is that all its information is guaranteed to be correct. For example, Facebook and Instagram both have ads and are USA-based, which is depicted in the vertical axis. On the other hand, the principal component analysis projection gives the impression that Instagram has more ads than Snapchat and both of them are not USA-based, which are both false claims.

However, Ganter and Glodeanu neither provide a method for the computation of ordinal factorizations nor does the visualization method transfer to larger datasets. These are two of the gaps in this realm that this chapter is meant to fill. First, we provide a method to compute ordinal two-factorizations if they exist. We couple this algorithm with a method to compute a subset of the incidence relation of large size such that it admits an ordinal two-factorization. This enables the computation of ordinal two-factorizations of arbitrary datasets. This process combined results in the algorithm ORD2FACTOR. Then, we propose a method for large datasets to greedily compute the factorizations. To this end, we propose ORDIFIND, a greedy algorithm to compute a complete ordinal factorization of a dataset leveraging fast algorithms from formal concept analysis. For the application of this greedy method, it is still necessary to demonstrate how to use it to discover and explain structure in such large datasets. We do so, by extending the existing theory of ordinal factor analysis into a tool to explain relationships in large binary datasets. Then, we demonstrate how to discover knowledge in some binary datasets using ordinal factor analysis by investigating different datasets.

5.2 Related Work

In this section, we consider methods that represent attributes in a low number of dimensions. First we discuss general approaches and then relate to factor analysis which was conducted in formal concept analysis, which is the realm in which our work is positioned.

5.2.1 Methods to Embed (Binary) Data

The two main reasons for the research of embedding high dimensional data into a lower number of dimensions are the following. First, embeddings are applied to escape the so-called curse of dimensionality [9], which is a phenomenon that causes distances to become less meaningful in higher dimensional spaces. Secondly, it provides a way to better visualize [66] complex data, as humans are better adapted to understand relationships in lower-dimensional spaces. For an extensive survey on dimensional reduction methods, we refer the reader to Espadoto et al. [45]. A commonly applied approach for embedding high dimensional data into lower dimensions is the principal component analysis [96], which is a method

	USA-based	premium	ads	private messages	group messages	mobile first	stories	timeline
Facebook	×		×	×	×		×	×
Instagram	×		×	×	×	×	×	×
Reddit	×	×	×	×				
Snapchat	×		×	×	×	×	×	
Telegram				×	×	×		
TikTok			×	×	×	×	×	×
Twitter	×	×	×	×	×	×		×
WeChat			×	×	×	×	×	
WhatsApp	×			×	×	×	×	
YouTube	×	×	×				×	

Figure 5.3: Running example: This dataset compares attributes of different social media platforms.

that minimizes the average squared distances from the data points to a line. It is often confused [70] with exploratory factor analysis [17], a technique that allows exploring a dataset by reconstructing underlying factors. The running example that we use to explain the theory of this section is the dataset in Figure 5.3 about different features and attributes of various social media platforms.

5.2.2 Boolean Factor Analysis

The canonical way to depict information in formal concept analysis is the concept lattice. Still, some approaches arrange data differently to make it more accessible. One of them is the Boolean factor analysis, on which research is conducted within [74, 73, 12, 11, 13] and outside [121] of formal concept analysis. We give a brief introduction to this method, as it gives rise to ordinal factorizations.

Definition 5.1 (Boolean Factor Analysis)
A formal context $\mathbb{K} = (G, M, I)$ can be disassembled into two *factorizing formal contexts* (G, F, I_{GF}) and (F, M, I_{FM}) such that

$$(g, m) \in I \iff (g, f) \in I_{GF} \text{ and } (f, m) \in I_{FM}$$

for some $f \in F$. Elements in F are called *Boolean factors*.

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17
1				×			×		×		×	×		×	×	×	×
2	×		×	×			×	×	×	×	×	×	×	×	×	×	×
3					×	×						×			×		×
4			×					×	×	×	×	×	×	×	×	×	×
5													×			×	×
6							×	×	×	×	×		×	×		×	×
7		×			×	×				×		×	×		×	×	×
8								×	×	×	×		×	×		×	×
9													×	×	×	×	×
10						×					×	×		×	×		

	a	b	c	d	e	f	g	h
f1	×		×	×	×	×	×	×
f2	×	×	×	×	×	×		×
f3	×		×	×	×	×	×	
f4	×		×	×	×		×	×
f5	×	×	×	×				
f6	×	×	×					
f7			×	×	×		×	×
f8			×	×	×	×	×	
f9			×	×	×		×	
f10			×	×	×	×		
f11			×				×	
f12	×		×					
f13				×	×	×		
f14							×	
f15	×							
f16				×	×			
f17				×				

Objects:	**Attributes:**
(1) Facebook	(a) USA-based
(2) Instagram	(b) premium
(3) Reddit	(c) ads
(4) Snapchat	(d) private messages
(5) Telegram	(e) group messages
(6) TikTok	(f) mobile first
(7) Twitter	(g) stories
(8) WeChat	(h) timeline
(9) WhatsApp	
(10) YouTube	

Figure 5.4: A conceptual Boolean factorization of the dataset from Figure 5.3 using 17 Boolean factors.

To grasp the relationship between Boolean factors and formal concepts, one can see, that by definition every Boolean factor corresponds to a subset of a formal concept. For each factor $f \in F$, let the *factorizing family* be the pair (A, B) with $A = \{g \mid (g, f) \in I_{GF}\}$ and $B = \{m \mid (f, m) \in I_{FM}\}$. Then, for each object $g \in A$ and each attribute $m \in B$ it holds that $(g, m) \in I$. Thus, it is sensible and possible to extend each factor such that its factorizing family is a formal concept.

Definition 5.2 (Conceptual Factorization)
A Boolean factorization where each factor corresponds to a formal concept is called a *conceptual factorization*.

A Boolean factorization with few factors can have two benefits. It compresses the size of a dataset and increases its understandability. However, the existence of a small factorization for a formal context is not guaranteed and the underlying decision problem computationally complex.

Theorem 5.3 (Belohlávek and Vychodil [11])
For a given k deciding if a factorization into k factors exists is NP*-complete.*

For our running example, there exists a conceptual factorization into 17 factors as depicted in Figure 5.4.

5.2.3 Ordinal Factor Analysis

In 2012, Ganter and Glodeanu [56] introduce the notion of ordinal factor analysis. Applications of the method are demonstrated in [58] where it is applied to some smaller medical datasets. In [10] the theory is lifted to the case of triadic incidence relations and in [59] into the setting of fuzzy formal contexts. Ganter and Glodeanu propose to group multiple Boolean factors into a single factor as follows.

Definition 5.4 (Many-valued Factor)
For a given formal context $\mathbb{K} = (G,M,I)$ with a pair of factorizing contexts (G,F,I_{GF}) and (F,M,I_{FM}), a set $E \subseteq F$ is called a *many-valued* factor of (G,M,I).

A group of Boolean factors with special interest for this thesis is the ordinal factor.

Definition 5.5 (Ordinal Factor)
An *ordinal factor* is defined as a many-valued factor E where for all elements $e_1, e_2 \in E$ with factorizing families (A_1,B_1) and (A_2,B_2), it holds that $A_1 \subseteq A_2$ or $A_2 \subseteq A_1$.

Similar to the case of Boolean factors, it is possible to extend each ordinal factor to a chain of formal concepts.

Definition 5.6 (Conceptual Ordinal Factorization)
An ordinal factor F is called *conceptual*, if there is a set of concepts $\{(A_i,B_i) \mid i \in [k]\}$ with $F = \bigcup_{i=1}^{k} A_i \times B_i$. An ordinal factorization is called *conceptual*, if all of the factors are conceptual.

By considering the elements in the incidence relation that are covered by the ordinal factor, it is possible to describe each ordinal factor as a Ferrers relations.

Definition 5.7 (Ferrers Relation)
A *Ferrers relation* F of cardinality k in the context (G,M,I) is a subset of $G \times M$ where for all $g,h \in G$ and $m,n \in M$ it holds that

$$(g,m) \in F \wedge (h,n) \in F \Rightarrow (g,n) \in F \vee (h,m) \in F$$

and $|F| = k$. We call a Ferrers relation F maximal in a formal context (G,M,I) if $F \subseteq I$ and there is no Ferrers relation $F' \subseteq I$ with $F \subsetneq F'$.

The maximal Ferrers relations correspond 1-to-1 to the maximal chains of the concept lattice, which describe the maximal conceptual ordinal factors. Because of this duality, we refer to both, maximal Ferrers relations and maximal chains of formal concepts, as *maximal ordinal factors*. We are interested in a set of ordinal factors that covers all incidences of the formal context, as it accurately describes the dataset.

Definition 5.8 (Complete Ordinal Factorization)
A *complete ordinal factorization* of width k of a formal context $\mathbb{K} = (G,M,I)$ is a set of Ferrers relations $F_1, \dots F_k$ such that $\bigcup_{i=1}^{k} F_i = I$.

Note that there is always a complete ordinal factorization with $\min(|G|,|M|)$ factors and this bound is sharp because of the nominal scale.

Consider once again the running example. It is not possible to do a factorization with two factors because no two of the three elements (YouTube, premium), (WhatsApp, mobile-first) and (Facebook, timeline) can appear in the same Ferrers relation. On the other hand, it is possible to generate a complete ordinal factorization into three factors as follows:

1) $f17 < f16 < f13 < f10 < f8 < f3 < f1$
2) $f15 < f12 < f6 < f5 < f2$
3) $f14 < f11 < f9 < f7 < f4 < f1$

Note, that every factor from the conceptual Boolean factorization in Figure 5.4 appears in at least one ordinal factor, and this is thus a complete conceptual ordinal factorization.

Ganter and Glodeau propose to depict the information of the dataset in a coordinate system using the ordinal factorizations. Every axis of this plot represents a single ordinal factor as follows. Each tick of such an axis is associated with a concept of the concept chain. The label for the tick contains the additional attributes that this concept gains compared to the previous ticks. The plot depicts

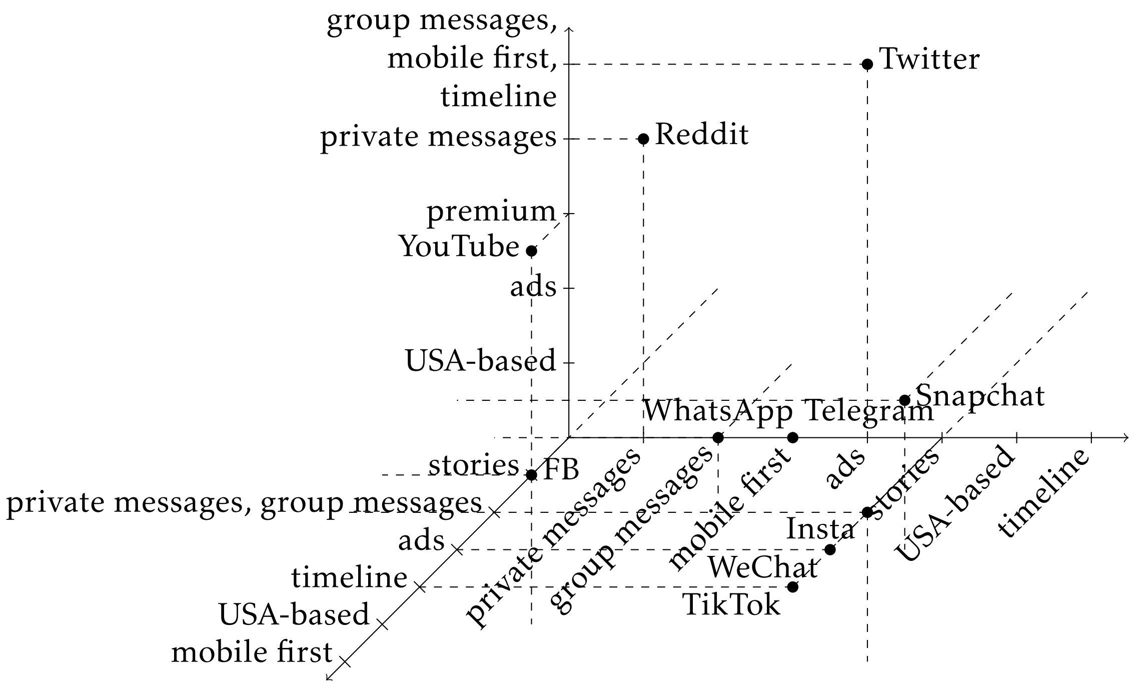


Figure 5.5: Complete ordinal factorization of the dataset from Figure 5.3. Compared to Figure 5.2, it does not miss information from the data, but is harder to read because it depicts a three-dimensional coordinate system in two dimensions.

the objects in each axis such that they are one tick before the first tick with an attribute they do not have. For our given factorization of the running example, the complete factorization would result in a three-dimensional coordinate system that can be hard to read, compare to Figure 5.5. Thus, restricting such a plot to the two largest ordinal factors is sensible, even though this implies that the information of the third factor is lost. In Figure 5.1 a principal component analysis projection is depicted which can be compared to the ordinal two-factorization plot in Figure 5.2.

For the analysis of ordinal factors, the incompatibility graph is of high importance.

Definition 5.9 (Incompatibility Graph)
The *incompatibility graph* of a formal context $\mathbb{K} = (G, M, I)$ is given by (I, E) with $\{(g,m),(h,n)\} \in E :\iff (g,n) \notin I \wedge (h,m) \notin I$. We call two pairs that share an edge *incompatible to each other*.

The incompatibility graph thus connects two pairs in the incidence if it is not possible for them to be in the same ordinal factor. The chromatic number of the incompatibility graph, i.e., the minimal number of colors needed to color the graph such that no two adjacent vertices share the same color is thus a lower bound for the number of ordinal factors required to factorize a context. For two-factorizable graphs this relationship is even stronger as the following theorem shows.

Theorem 5.10 (Doignon et al. [26])
A formal context is completely factorizable into two ordinal factors if and only if the incompatibility graph is bipartite.

This allows us to check in polynomial time, whether a formal context admits an ordinal two-factorization. Provided $\mathsf{P} \neq \mathsf{NP}$, such a polynomial-time check is not possible for factorizations into more ordinal factors as the minimal number of factors is closely related to the order dimension.

Theorem 5.11 (Theorem 46 [57])
Let $\mathbb{K} = (G, M, I)$ be a formal context. $\underline{\mathfrak{B}}(\mathbb{K}^c)$ has order dimension at most k if and only if $\mathbb{K}$ has an ordinal factorization into k ordinal factors.

As it is NP-complete to decide, whether the order dimension of an ordered set is k for $k \geq 3$ [119] and the order dimension of $(X, \leq)$ and $\mathfrak{B}(X, X, \leq)$ are equal it is also NP-complete to decide whether there is a complete ordinal factorization into k factors for $k \geq 3$.

It follows directly from the definition of the Ferrers relation that its complement is once again a Ferrers relation.

Proposition 5.12
Given a formal context (G, M, F) with F being a Ferrers relation. Then the formal context $(G, M, (G \times M) \setminus F)$ is also a Ferrers relation.

One important observation that we should add at this point is the relationship between Ferrers relations and ordinal scales. By definition, a Ferrers relation with k concepts contains an ordinal scale of length k. On the other hand is each ordinal scale a Ferrers relation. Thus, discovering large Ferrers relations can be related to discovering large ordinal scales and the problem discussed in this chapter can be seen dual to the one discussed in the previous chapter.

5.3 Ordinal Two-Factorizations

Ordinal two-factorizations provide insight into the structure of the dataset, as they allow for the visualization method that we can see in Figure 5.2. Still, it is not yet explored how to compute such factorizations efficiently. In this section, we fill in the missing pieces by providing an algorithm that computes an ordinal two-factorization for a given dataset where such a factorization exists, i.e., one with a bipartite incompatibility graph. Furthermore, we investigate the problem of computing a maximal subset of the incidence relation that admits an ordinal two-factorizaition.

5.3.1 Disjoint Ordinal Two-Factorizations

If it is possible, it is preferred to compute two ordinal factors that are disjoint, i.e., two Ferrers relations that do not share pairs of the incidence relation. However, as the formal context in Figure 5.6 shows, this is not always possible. The example is due to Das et al. [23] where they characterize the formal contexts that are two-factorizable as interval digraphs. The incompatibility graph of the example consists of two connected components, one being the graph induced by $\{(6, f)\}$ and the other being $I \setminus \{(6, f)\}$. We can assign the bipartition classes of the second component to factor 1 and 2 without loss of generality as shown on the right side of the figure. But the incidence pair then $(6, f)$ has to be in both factors, in factor 1 because of the incidences $(6, e)$ and $(2, f)$ and in factor 2 because of the incidences $(5, f)$ and $(6, b)$.

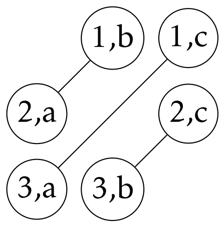
Generally, we can deduce from the incompatibility graph incidence pairs that cannot appear in the same ordinal factor. Still, we note that even in cases where disjoint two-factorizations do exist, the bipartition classes of the incompatibility graph do not necessarily give rise to an ordinal two-factorization. To see this, refer to Figure 5.7. The incompatibility graph (middle) of the formal context (left) consists of three components. An assignment of the incompatibility graph to bipartition classes can be seen on the right, however the elements $(2, a)$ and $(3, b)$ of factor 2 would imply, that the element $(3, a)$ also has to be in factor 2.

	a	b	c	d	e	f	g
1				×	×	×	×
2						×	×
3							×
4	×						
5	×					×	
6	×	×			×	×	×
7	×	×	×			×	

	a	b	c	d	e	f	g
1				1	1	1	1
2						1	1
3							1
4	2						
5	2					2	
6	2	2			1	×	1
7	2	2	2			2	

Figure 5.6: Example of a context with a maximal bipartite subgraph that does not give rise to an ordinal two-factorization. This example is due to Das et al. [23].

	a	b	c
1		×	×
2	×		×
3	×	×	



	a	b	c
1		1	2
2	2		1
3	1	2	

Figure 5.7: *Left:* The formal context of a contranominal scale. *Middle:* its comparability graph. *Right:* A bipartition of the transitive comparability graph that does not give rise to an ordinal two-factorization.

However, this element is incompatible to element $(3,c)$, which is also in factor 2 by the assignment of the incompatibility graph. Thus, such an assignment does not always result in a valid ordinal two-factorization.

On the other hand, if the incompatibility graph is connected and bipartite, any assignment of the elements to bipartition classes of the incompatibility graph generates a valid ordinal two-factorization, as the following shows.

Theorem 5.13
Let $\mathbb{K}$ be a formal context with a connected and bipartite incompatibility graph. Then there are two unique disjoint factors F_1 F_2 that factorize $\mathbb{K}$. The sets F_1 and F_2 correspond to the bipartition classes of the incompatibility graph.

Proof. Follows from Theorem 5.10 and the fact, that two elements that are connected by an edge in the incompatibility graph cannot be in the same ordinal factor. □

We can further characterize the elements that can be in both bipartition classes as follows:

Theorem 5.14
Let $\mathbb{K} = (G, M, I)$ be a formal context with bipartite incompatibility graph (I, E). Let F_1, F_2 be an ordinal two-factorization of $\mathbb{K}$. For all elements $(g, m) \in F_1 \cap F_2$ it holds that $\{(g, m)\}$ is a connected component in (I, E).

Proof. Assume not, i.e., there is an element $(g, m) \in F_1 \cap F_2$ that is not its own component. Then, there has to be some element $(h, n) \in I$ that is incompatible to (g, m), i.e., $(g, n) \notin I$ and $(h, m) \notin I$. But then (h, n) can be in neither F_1 nor F_2 which contradicts the definition of a Ferrers relation. □

Thus, only the isolated elements of the incompatibility graph can be in both bipartition classes. The following further characterize the isolated elements, as we show that they can always be in both factors, which then fully describes the potential intersection of the two ordinal factors.

Theorem 5.15
Let $\mathbb{K} = (G, M, I)$ be a formal context with a two-factorization F_1, F_2. Let C be the set of all elements of I that are incompatible to no other element. Then $F_1 \cup C$, $F_2 \cup C$ is also an ordinal two-factorization.

Proof. Let $G_i = F_i \setminus C$ and $\tilde{F}_i = F_i \cup C$ for $i \in \{1, 2\}$ Assume the statement is not true, i.e., either $\tilde{F}_1$ or $\tilde{F}_2$ is no ordinal factor. Without loss of generality, let $\tilde{F}_1 = F_1 \cup C$ be no ordinal factor. Then, there are two elements $(g, m), (h, n) \in \tilde{F}_1$ such that $(g, n) \notin \tilde{F}_1$ and $(h, m) \notin \tilde{F}_1$. As F_1 is an ordinal factor, at least one of the two elements has to be in C, let without loss of generality $(g, m) \in C$. We now do a case distinction whether one or both of them are in C.
Case 1. Let first $(g, m) \in C$ and $(h, n) \notin C$. One of the elements (g, n) or (h, m) has to be in I, otherwise (g, m) and (h, n) are incompatible, without loss of generality, let $(h, m) \in I$. As $(h, m) \notin C$, it has to hold that $(h, m) \in G_2$ and there has to be some $(x, y) \in G_1$ with $(h, y) \notin I$ and $(x, m) \notin I$. As $(x, y) \in G_1$, $(h, n) \in G_1$, and $(h, y) \notin I$ and F_1 is an ordinal factor, $(x, n) \in F_1$. As $(g, m) \in C$ it is incompatible with no element and thus not incompatible with (x, n) in particular, but $(x, m) \notin I$, it holds that $(g, n) \in I$. The element (g, n) has to be in G_2, as otherwise (g, m) and (h, n) are not incompatible. Thus, there also has to be some element in $(a, b) \in G_1$ with $(a, n) \notin I$ and $(g, b) \notin I$. As F_1 is an ordinal factor and $(x, n) \in F_1, (a, b) \in F_1$ and $(a, n) \notin F_1$, the element $(x, b) \in F_1$. But then (x, b) is incompatible to (g, m) which is a contradiction to (g, m) being in C.
Case 2. Let $(g, m) \in C$ and $(h, n) \in C$. Either $(g, n) \in I$ or $(h, m) \in I$, let without loss of generality $(g, n) \in I$. Then it has to be hold more specifically that in $(g, n) \in G_2$. Thus, there is some element $(x, y) \in G_1$ with $(h, y) \notin I$ and $(x, m) \notin I$. Because (x, y) has to be compatible with (h, n), it has to hold that $(x, n) \in I$. On the other hand, (x, n) has to be compatible with (g, m), thus $(g, n) \in I$. It then has to hold that $(g, n) \in G_2$ and thus some element $(a, b) \in G_1$ has to exist with $(g, b) \notin I$ and $(a, n) \notin I$. As $(x, y) \in F_1$ and $(a, b) \in F_1$ and F_1 is an ordinal factor, either $(a, y) \in F_1$ or $(x, b) \in F_1$. If $(a, y) \in F_1$, it would be incompatible to (g, m), if $(x, b) \in F_1$ it would be incompatible to (h, n). Both would be a contradiction to the respective element being in C.

This finishes a complete characterization of the non-disjoint part of ordinal factors. Therefore, a partition of the incidence as follows always exists, if the context is ordinal two-factorizable.

Proposition 5.16
Let $\mathbb{K} = (G,M,I)$ be a two-factorizable formal context with (I,E) its incompatibility graph. Then there is a partition of I into F_1, F_2, C with $C = \{D \mid D$ connected component of $(I,E), |D| = 1\}$ and $F_1 \cup C$ and $F_2 \cup C$ are Ferrers relations.

Proof. This directly follows from the previous theorem. Let $\tilde{F}_1, \tilde{F}_2$ be a two-factorization and $C = \{D \mid D$ connected component of $(I,E), |D| = 1\}$. Then the partition is given by $\tilde{F}_1 \setminus C$, $\tilde{F}_2 \setminus C$, and C. □

5.3.2 An Algorithm for Ordinal Two-Factorizations

Now, we propose an algorithm to compute ordinal two-factorizations if they exist. As we saw in the last section, the bipartition classes of the incompatibility graph do not directly give rise to an ordinal two-factorizations. An important observation [57, Thm.46] is that a formal context can be described by the intersection of two Ferrers relations, if and only if its corresponding concept lattice can be described as the intersection of two linear orders, i.e., if it has order dimension two. As the complement of a Ferrers relation is once again a Ferrers relation, a formal context is two-factorizable if and only if the concept lattice of its complement context has order dimension two. We leverage this relationship with the following theorem, to explicitly compute the ordinal two-factorization.

Theorem 5.17
Let $\mathbb{K} = (G,M,I)$ be a formal context. Let $(\mathfrak{B}, \leq) = \underline{\mathfrak{B}}(\mathbb{K}^c)$. If $\mathbb{K}$ is two-factorizable, then $\leq$ is two-dimensional and there is a conjugate order $\leq_c$. The sets

$$F_1 = \{(g,m) \in G \times M \mid \nexists (A,B),(C,D) \in \mathfrak{B}, g \in A, m \in D, ((A,B),(C,D)) \in (\leq \cup \leq_c)\}$$

and

$$F_2 = \{(g,m) \in G \times M \mid \nexists (A,B),(C,D) \in \mathfrak{B}, g \in A, m \in D, ((A,B),(C,D)) \in (\leq \cup \geq_c)\}$$

give rise to an ordinal factorization of $\mathbb{K}$.

Proof. We have to show that F_1 and F_2 are Ferrers relations and $F_1 \cup F_2 = I$. We first show that F_1 is a Ferrers relation. By definition $\lessdot := \leq \cup \leq_c$ is a chain ordering all formal concepts of $\mathfrak{B}(K^c)$. Let (g,m) and (h,n) be two pairs in F_1. Assume that $(g,n) \notin F_1$ and $(h,m) \notin F_1$. Then there have to be two concept (A_1,B_1) and (A_2,B_2) with $g \in A_1$, $n \in B_2$ such that $(A_1,B_1) \lessdot (A_2,B_2)$. Similarly, there have to be two concepts (A_3,B_3) and (A_4,B_4) with $h \in A_3$, $m \in B_4$ such that $(A_3,B_3) \lessdot (A_4,B_4)$. For the concepts (A_2,B_2) and (A_3,B_3) it holds that $(A_2,B_2) \lessdot (A_3,B_3)$, as

$(A_2,B_2) \neq (A_3,B_3)$ and $(A_3,B_3) \nless (A_2,B_2)$ because $(g,m) \notin F_1$. By the same argument, $(A_4,B_4) < (A_1,B_1)$. Thus, $(A_1,B_1) < (A_2,B_2) < (A_3,B_3) < (A_4,B_4) < (A_1,B_1)$, which would imply that this three concepts are equal and is thus a contradiction. This proves that F_1 is a Ferrers relation. The argument to shows that F_2 is a Ferrers relation is dual. We now show that $F_1 \cup F_2 = I$. Let $(g,m) \in F_1 \cup F_2$, without loss of generality let it be an element of F_1. Then there are no two concepts $(A,B),(C,D) \in \mathfrak{B}(\mathbb{K}^c)$, with $g \in A$, $m \in D$ and $((A,B),(C,D)) \in (\leq \cup \leq_c)$. Consider the concept (g'',g') with the derivation from the complement context. By definition $((g'',g'),(g'',g')) \in (\leq \cup \leq_c)$ and thus $m \notin g'$ when using the derivation from the complement context, i.e., $(g,m) \notin I^c$. But then, $(g,m) \in I$. Now, let $(g,m) \in I$ and assume that $(g,m) \notin F_1$. Let (A,B) and (C,D) be arbitrary concepts of $\mathfrak{B}(\mathbb{K}^c)$ with $g \in A$ and $m \in D$. As $(g,m) \in I$, it is not in the incidence of $\mathbb{K}^c$ and thus (A,B) and (C,D) are not comparable with $\leq$. As $(g,m) \notin F_1$ it holds $(A,B) \nleq_c (C,D)$ and as they are incomparable with $\leq$ it has to hold that $(A,B) \geq_c (C,D)$. Thus $(g,m) \in F_2$. This shows that $F_1 \cup F_2 = I$ and thus concludes the proof. □

This proof gives rise to the routine in Algorithm 5.6 where this information is used to compute an ordinal two-factorization of the formal context. It computes the two sets F_1 and F_2 from the previous theorem. To do so, it has to be paired with an algorithm to compute the concept lattice. An algorithm that is suitable for this due to Lindig [86], as it computes the covering relation together with the concept lattice. Furthermore, an algorithm for transitive orientations [60] is required to compute the conjugate order. As we will discuss later, for both these algorithms the runtime is not critical, as we are interested in ordinal two-factorizations of small formal contexts. Modern computers are easily able to deal with such contexts. Still, if suitable supporting algorithms are chosen, this results in a polynomial-time algorithm that computes an ordinal two-factorization for a given two-factorizable formal context.

5.3.3 Maximal Ordinal Two-Factorizations

In this section, we propose an algorithm to compute ordinal two-factorizations for a given dataset that covers a large part of the incidence relation.

Definition 5.18 (Maximal Ordinal Two-Factorizations)
Let $\mathbb{K} = (G,M,I)$ be a formal context. A *maximal ordinal two-factorization* of $\mathbb{K}$ is a set of two Ferrers relations $F_1,F_2 \subseteq I$ such that there are no Ferrers relations $\tilde{F}_1,\tilde{F}_2 \subseteq I$ with $|\tilde{F}_1 \cup \tilde{F}_2| \geq |F_1 \cup F_2|$.

Algorithm 5.6 Compute Ordinal Two-Factorization

Input: Ordinal Two-Factorizable Formal Context $\mathbb{K} = (G, M, I)$
Output: Ordinal Two-Factorization F_1, F_2

```
def two_factor($G,M,I$):
    $(\mathfrak{B},\leq)$ = $\underline{\mathfrak{B}}(\mathbb{K}^c)$
    $(\mathfrak{B},E)$ = co_comparability_graph($\mathfrak{B},\leq$)
    $\leq_c$ = transitive_orientation($\mathfrak{B},E$)
    $\lessdot_1$ = $\leq \cup \leq_c$
    $\lessdot_2$ = $\leq \cup \geq_c$
    for $i$ in $\{1,2\}$:
        $L_i$ = $\{\}$
        $\tilde{A}$ = $\{\}$
        for $(A,B)$ in $\mathfrak{B}$ ordered by $\lessdot_i$:
            $\tilde{A}$ = $\tilde{A} \cup A$
            $L_i$ = $L_i \cup (\tilde{A} \times B)$
        $F_i$ = $(G \times M) \setminus L_i$
    return $F_1,F_2$
```

While there are various thinkable ways, how one could define maximal ordinal two-factorizations, we have chosen to do so by minimizing the size of not-covered incidence relation pairs, as each element in the incidence relation can be seen as a data point that would otherwise be lost. This definition also aligns with a suggestion of Ganter from his textbook where he recommends to maximize the size of the union of the two Ferrers relations [54]. Thus, the problem we approach in this section is formalized as follows:

Problem 5.19 (Maximal Ordinal Two-Factorization Problem)
Given: A formal context (G, M, I).
Requested: Two Ferrers relations $F_1, F_2 \subseteq I$.
Optimization: Maximize $|F_1 \cup F_2|$.

Note, that this problem is well-defined, as a single element of the incidences is a Ferrers relation by itself.

5.3.4 Maximal Ordinal Two-Factorizations are Hard

First, we investigate the computational complexity of the Maximal Ordinal Two-Factorization Problem. To do so, we need the related problem to compute that will also play a role in Chapter 7.

Problem 5.20 (Two-Dimension Extension Problem)
Given: An ordered set $(X,\leq)$ and a $k\in\mathbb{N}$.
Requested: Is there a set $\tilde{\leq}\supseteq\leq$ such that is an order and $(X,\tilde{\leq})$ has order dimension two and $|\tilde{\leq}|-|\leq|=k$.

The problem requires finding the minimum number of pairs that need to be added to a relation in order to make it two-dimensional, and it is of high complexity as shown by the following.

Theorem 5.21 (Felsner and Reuter [48])
Deciding the Minimal Two-Dimension Extension Problem is NP*-complete.*

To investigate the complexity of computing maximal ordinal two-factorizations, we consider a formulation of the problem as a decision problem. Thereby, we ask for the existence of a factorization that covers all except k incidence pairs.

Problem 5.22 (Ordinal Two-Factorization Problem)
Given: A formal context (G,M,I) and a $k\in\mathbb{N}$.
Requested: Is there a formal context $(G,M,\tilde{I})$ with $\tilde{I}\subseteq I$ and $|I|-|\tilde{I}|=k$ that has an ordinal two-factorization?

The relation between the Two-Dimension Extension Problem and the Ordinal Two-Factorization Problem gives rise to the computational complexity of computing a two-factorization.

Lemma 5.23
There is a polynomial-time reduction from the Two-Dimension Extension Problem to the Ordinal Two-Factorization Problem .

Proof. Let $(X,\leq)$ and $k\in\mathbb{N}$ be an instance of the Two-Dimension Extension Problem.
Claim: The problem has a solution if and only if Ordinal Two-Factorization Problem $(X,X,\not\leq)$ with k has a solution.

Let $(X,\leq)$ be an ordered set with a two-dimension-extension C of size k. Let L_1, L_2 be a realizer, i.e., two linear extensions of $\leq\cup C$ with $L_1\cap L_2=\leq\cup C$. Then the relations $F_1:=(X\times X)\setminus L_1$ and $F_2:=(X\times X)\setminus L_2$ are Ferrers relations. Furthermore, $F_1\subseteq\not\leq$ and $F_2\subseteq\not\leq$ by definition. Assume that $F_1\cup F_2\cup C\neq\not\leq$. Then there has to be a pair $a,b\in X$ with $a\not\leq b$ and $(a,b)\notin C$. Then $(b,a)\in L_1$, or $(b,a)\in L_2$, or both, without loss of generality let $(b,a)\in L_1$. But this implies that $(a,b)\in L_1$ which is a contradiction.

Now, let for the formal context $(X, X, \not\leq)$ be C a set of cardinality k such that there are two Ferrers relations F_1, F_2 with $|F_1 \cup F_2 \cup C| = |\not\leq|$. Now, let $L_1 = (X \times X) \setminus F_1$ and $L_2 = (X \times X) \setminus F_2$. Then it holds that $L_1 \cap L_2 = \leq \cup C$. L_1 and L_2 are supersets of $\leq$ and Ferrers relations, thus they are transitive, reflexive and for all elements $a, b \in X$ it holds that $a, b \in L_i$ or $b, a \in L_i$. By definition for both $i \in \{1, 2\}$ there is a $\tilde{L}_i \subseteq L_i$ that has all these properties and is also antisymmetric. The existence of $\tilde{L}_i$ follows from placing a linear order on each of the equivalence classes of L_i. Both $\tilde{L}_1$ and $\tilde{L}_2$ are linear extensions of $\leq$ and $|\tilde{L}_1 \cap \tilde{L}_2| \leq |L_1 \cap L_2| = |\leq| + k$.

This proves the claim. Thus, we reduced the TWO-DIMENSION EXTENSION PROBLEM to the ORDINAL TWO-FACTORIZATION PROBLEM.

Lemma 5.24
Validation of a solution of the ORDINAL TWO-FACTORIZATION PROBLEM can be done in polynomial time.

Proof. Given a formal context (G, M, I) and a set $C \subseteq I$ of size k, to check whether $(G, M, I \setminus C)$ admits an ordinal two-factorization is equivalent to the check whether the incompatibility graph of $(G, M, I \setminus C)$ is bipartite because of Theorem 5.10. □

Thus, for the ORDINAL TWO-FACTORIZATION PROBLEM the following holds.

Theorem 5.25
Solving the ORDINAL TWO-FACTORIZATION PROBLEM is NP-*complete.*

Proof. Follows from Lemmas 5.23 and 5.24. □

5.3.5 Maximal Bipartite Subgraphs are Not Sufficient

The structure of the incompatibility graph provides an interesting foundation to compute ordinal two-factorizations. It seems to be a tempting idea to compute the maximal induced bipartite subgraph of a formal context. The vertex set that induces such a maximal bipartite subgraph could then be used for the ordinal two-factorization. However, it turns out a bipartite subgraph does not always give rise to an ordinal two-factorization as Figure 5.8 demonstrates. Its incompatibility graph has an odd cycle, and it is thus not bipartite. This makes the formal context not two-factorizable. An inclusion-minimal set that can be removed to make it bipartite is given by

$$\begin{aligned} C = \{&(6,j),(4,n),(7,p),(18,p),(6,p),(6,n),(12,k),(10,g),(6,g),(5,p),(2,i),\\ &(4,p),(12,m),(3,i),(12,h),(1,p),(2,q)\}. \end{aligned}$$

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
1		×	×	×		×		×		×	×	×	×	×	×	×		
2	×		×	×	×	×	×	×	×	×	×	×	×	×	×		×	
3	×	×		×	×	×	×	×	×	×	×	×	×	×	×			×
4	×	×	×		×	×			×	×		×		×		×	×	×
5	×	×	×	×		×		×		×	×	×	×	×	×	×	×	×
6	×	×	×	×	×		×			×		×		×		×	×	×
7	×	×	×	×	×	×		×	×	×	×	×	×	×	×	×	×	×
8	×	×	×	×	×	×	×		×	×		×		×		×	×	×
9	×	×	×	×	×	×	×	×		×	×	×	×	×	×	×	×	×
10	×	×	×	×	×	×	×	×	×		×	×			×		×	×
11	×	×	×	×	×	×	×	×	×	×		×		×		×	×	×
12	×	×	×	×	×	×	×	×	×	×	×		×				×	×
13	×	×	×	×	×	×	×	×	×	×	×	×		×	×	×	×	×
14	×	×	×	×	×	×	×	×	×	×	×	×	×		×		×	×
15	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×	×	×
16	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×	×
17	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×
18	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	

Figure 5.8: Example of a formal context with a maximal bipartite subgraph that does not give rise to an ordinal two-factorization.

However, for the formal context $(G,M,I \setminus C)$ it once again holds that it is not two-factorizable as its incompatibility graph contains once again an odd cycle. This is possible as new incompatibilities can arise from the removal of incidence pairs.

5.3.6 Computing Maximal Ordinal Two-Factorizations

In the last sections, we did structural investigations on ordinal two-factorizations and provided an algorithm to compute them. We now use this to compute a large ordinal two-factorization. To this end, we propose the algorithm ORD2FACTOR in Algorithm 5.7. Thereby, the induced bipartite subgraph of the incompatibility graph is computed. As new incompatibilities can arise by the removal of crosses, as noted previous section, we might have to repeat this procedure.

The induced bipartite subgraph can be computed using the methods from Chapter 3. We are not aware of a formal context where using the SAT-sovler approach to solve the maximal bipartite subgraph problem requires a second repetition of the algorithm. Therefore, we formulate the following open question.

Algorithm 5.7 ORD2FACTOR to Compute Large Ordinal Two-Factorization

Input: Formal Context (G,M,I)
Output: Ordinal Factors F_1 and F_2

```
def Ord2Factor($G,M,I$):
    $(I,E)$ = incompatibility_graph($G,M,I$)
    while $(I,E)$ not bipartite:
        $I$ = maximal_bipartite_inducing_vertex_set($I,E$)
        $(I,E)$ = incompatibility_graph($G,M,I$)
    return two_factors($G,M,I$)
```

Open Problem 5.26
Is there a formal context (G,M,I), such that a maximal set $\tilde{I} \subseteq I$ that induces a bipartite subgraph on the incompatibility graph does not give rise to a two-factorizable formal context $(G,M,\tilde{I})$?

This open problem is of special interest, because it would allow our approach to compute globally maximal two-factorization if combined with the SAT-solver approach from Chapter 3, as the following shows.

Theorem 5.27
Let $\mathbb{K} = (G,M,I)$ be a formal context and $\tilde{I}$ the subset of I that induces a maximal bipartite subgraph on the incompatibility graph. If the formal context $\mathbb{K} = (G,M,\tilde{I})$ admits an ordinal two-factorization, its factors are the maximal ordinal factors of $\mathbb{K}$.

Proof. Assume there are two ordinal factors F_1 and F_2 of $\mathbb{K}$ and $|F_1 \cup F_2| > |\tilde{I}|$. But then the context $(G,M,F_1 \cup F_2)$ is a two-factorizable and thus the graph induced by $F_1 \cup F_2$ on the incompatibility graph is bipartite, a contradiction. □

5.4 Greedy Ordinal Factorizations

In large bipartite graphs, it is hard to discover and analyze structure. In this section we propose a way to do so by computing complete ordinal factorizations, which precisely represent the original dataset. Based on such an ordinal factorization, we provide a way to discover and explain relationships between different objects and attributes in the dataset. However, computing even just one ordinal factor of high cardinality is computationally complex. We thus propose the greedy algorithm ORDIFIND in this section. This algorithm extracts ordinal factors using already existing fast algorithms developed in formal concept analysis. Then, we leverage ORDIFIND to propose a comprehensive way to discover relationships in the dataset. We furthermore introduce a distance measure based on the representation emerging from the ordinal factorization to discover similar objects.

5.4.1 Greedy Ordinal Factorizations are Hard

In the following, we introduce and investigate greedy ordinal factorizations. In contrast to the previous section, these factors are not computed concurrently, but they are achieved by iteratively computing the largest ordinal factor remaining in the dataset.

Definition 5.28 (Greedy Ordinal Factorization)
Let (G,M,I) be a formal context. A set of ordinal factors $F_1,\dots,F_k$ is called *greedy*, if for each $i \in \{1,\dots,k\}$ there is no factor $\tilde{F_i}$ with $|\tilde{F_i} \setminus \{F_1 \cup \dots \cup F_{i-1}\}| > |F_i \setminus \{F_1 \cup \dots \cup F_{i-1}\}|$.

Repeatedly computing a Ferrers relation, which covers the maximum possible uncovered part of the incidence, results in such a factorization. A complete ordinal factorization arises by repeating this process until every part of the incidence relation is covered. Still, it turns out that even deciding on the size of the first ordinal factor of this greedy process is an NP-complete problem. To see this, consider an auxiliary Lemma from Yannakakis [118]. This Lemma refers to the bipartite graphs corresponding to Ferrers relations as chain graphs. Adapted to the notions of this work the statement is the following.

Lemma 5.29 (Yannakakis [118])
Let $\mathbb{K} = (G,M,I)$ be a formal context and k a natural number. It is NP-*complete to decide whether there is a set $\tilde{I} \subseteq G \times M$ of size k such that $I \cup \tilde{I}$ is a Ferrers relation.*

The complement of a Ferrers relation is once again a Ferrers relation. Thus, it follows immediately that deciding on the number of pairs that have to be removed from a binary relation to make it a Ferrers relation is equally computationally complex.

Corollary 5.30
Let $\mathbb{K} = (G,M,I)$ be a formal context and k a natural number. It is NP-*complete to decide whether there is a set $\tilde{I} \subseteq I$ of size k such that $I \setminus \tilde{I}$ is a Ferrers relation.*

Thus, even computing the first greedy factor entails solving an NP-complete task which makes the problem of computing a greedy ordinal factorization computationally complex.

Algorithm 5.8 Compute Maximal Ferrers Relation

Input: Concept lattice with covering relation $(\mathfrak{B}, \prec)$
Covered E
Output: Maximal Ferrers Relation F

```
$F_{(A,B)} = \emptyset \;\forall (A,B) \in \mathfrak{B}$
$L = linear\_extension(\mathfrak{B}, \prec)$
for $(A_1, B_1)$ in $L$:
    for $(A_2, B_2) \in \mathfrak{B}$ with $(A_2, B_2) \prec (A_1, B_1)$:
        $\tilde{F}_{(A_1,B_1)} = F_{(A_2,B_2)} \cup (A_1 \times B_1)$
        if $|\tilde{F}_{(A_1,B_1)} \setminus E| \geq |F_{(A_1,B_1)} \setminus E|$:
            $F_{(A_1,B_1)} = \tilde{F}_{(A_1,B_1)}$
return $F_{\max(\mathfrak{B},\prec)}$
```

5.4.2 OrdiFind to Compute Greedy Ordinal Factorizations

In this section, we propose ORDIFIND (Algorithm for ORDinal Factors IN binary Data), an algorithm to compute greedy ordinal factorization. As discussed in the last section, this problem is of high computational complexity. Thus, the best algorithm we can hope to find will most likely not run in polynomial time with respect to the input size. Still, it is possible to leverage fast concept lattice algorithms from formal concept analysis. Especially Lindig's algorithm [86] which can be equipped with speed-up techniques [80] is useful. This algorithm computes the covering relation of a formal context in a reasonable time. The main advantage of this approach is that the exponential-time task of computing the concept lattice is executed only once. After this, the algorithm has linear time complexity in the size of the covering relation of the concept lattice. Note, that the size of this covering relation may still be exponential in the size of the input data.

The routine described in Algorithm 5.8 lays the foundation for ORDIFIND. It takes the covering relation of a concept lattice and a set E as its input. The routine then computes an ordinal factor that covers a maximal number of incidences in $I \setminus E$. Thus, the output of the routine is an ordinal factor F, in its Ferrers relation form, such that there is no ordinal factor $\tilde{F}$ with $|\tilde{F} \setminus E| > |F \setminus E|$. The main idea of the routine is the following. The concept lattice of a formal context is iterated from the bottom to the top in order of a linear extension, i.e., in a linear order compatible with the concept lattice. Thus, each concept is not processed before the same process finishes for all smaller ones. After each concept is processed, the ordinal factor covering a maximal number of incidences of $I \setminus E$ with this specific concept (A,B) as a top concept is in the variable $F_{(A,B)}$. Therefore, after

Algorithm 5.9 Naive Complete Ordinal Factorization

Input: Formal context (G,M,I)
Covering relation $(\mathfrak{B},\prec)$
Output: Greedy ordinal factorization $F_1,\dots,F_k$

```
$E = \emptyset$
$i = 1$
while $E \neq I$:
    $F_i = max\_ferrers((\mathfrak{B},\prec),E)$
    $E = E \cup F_i$
    $i = i+1$
return $F_1,\dots,F_i$
```

every element is processed, the variable corresponds to the top concept $F_{\max(\mathfrak{B},\prec)}$ contains the factor excluding with the highest number of incidences from $I \setminus E$. For a set of already computed factors, it is possible to compute the ordinal factor containing a maximal number of uncovered incidences by choosing the union of those factors as E. Iteratively repeating this procedure results in a complete greedy factorization of the formal context as described in Algorithm 5.9.

Consequently, the following lemma concerning the overall runtime of the algorithm follows.

Lemma 5.31
For a formal context with k concepts and given covering relation, it is possible to compute a greedy ordinal factorization consisting of r ordinal factors in $\mathcal{O}(rk^2)$.

Proof. The runtime of Algorithm 5.8 is bounded from above by the size of the covering relation. The covering relation is bounded from above by the number of concept pairs. The loop in Algorithm 5.9 is repeated r times. □

The algorithm described above computes a greedy ordinal factorization. For the usefulness of this algorithm, it is crucial to investigate how good the factorization is, i.e., how it performs compared to an ordinal factorization that minimizes the number of factors. We investigate this in the following Lemma.

Lemma 5.32
Let $\mathbb{K} = (G,M,I)$ that can be optimally decomposed into k ordinal factors. Let $F_1,\dots,F_r$ be a greedy ordinal factor decomposition. Then

$$r \leq k \log_e |I|.$$

Proof. By pigeon-hole principle, it holds that

$$|F_i| \geq (|I| - |F_1 \cup \ldots \cup F_{i-1}|)/k$$

for all $i \in \mathbb{N}$. Thus, for $r > k \log_e |I|$

$$|I| - |F_1| - \cdots - |F_r| \leq |I|(1 - \frac{1}{k})^r \leq |I| e^{-r/k} < 1,$$

implying that all incidences are covered after r repetitions. □

The naive algorithm, proposed in Algorithms 5.8 and 5.9, reprocesses each concept in every step for each computation of a new factor. However, especially in the later and smaller factors, the algorithm does not modify all concepts in the lattice. Identifying concepts not touched by the algorithm enables a speed-up technique as the value of those can be saved between the steps. To do so, we have to modify how the naive algorithm stores the computed information. Instead of saving all incidences of the ordinal factor at each concept in the lattice, we only store their size. Additionally, we store for each concept in the factor its predecessor. The factor is then extracted by trailing the predecessors, starting with the top concept. Now, not every factor contains every attribute. Thus, after a factor with missing attributes was computed, only concepts that have a greater concept with an attribute in it have to be recomputed. The speed-up thus emerges from just recomputing only the changed concepts.

To determine which concepts are to recompute, we store the attributes of the lastly computed factor in some set R. We then use the set R furthermore to determine which concepts to iterate next. Instead of enumerating the concepts in the order of a previously fixed linear extension, the smallest attribute set of R is always the next one to iterate. Then it is removed from R. To enable this extraction, we implement R as a heap. As the size of the attribute sets of concepts increases the smaller the concepts in the concept lattice, a linear order compatible with the concept lattice is respected. The proposed algorithm OrdiFIND in Algorithm 5.10 uses the ideas from this paragraph to speed up the first two algorithms.

5.4.3 Relationship between Factors and Objects

Let (G, M, I) be a formal context with an ordinal factorization $F_1, \ldots, F_k$. For a fixed factor $F_i = (A_1, B_1), \ldots, (A_j, B_j)$ and an object g call the position of g in F_i the maximal r, such that $B_r \subseteq g'$. Because F_i is a chain of concepts, all attribute sets of

Algorithm 5.10 ORDIFIND to Compute a Complete Ordinal Factorization

Input: Formal context (G,M,I)
Covering relation $(\mathfrak{B},\prec)$
Output: Greedy ordinal factorization $F_1,\ldots,F_k$

```
$E = \emptyset$
$i = 1$
$R = \min(\mathfrak{B}\prec)$
while $E \neq I$:
    $b_{(A,B)} = \perp \ \forall (A,B) \in \mathfrak{B}$
    $s_{(A,B)} = \emptyset \ \forall (A,B) \in \mathfrak{B}$
    while $R \neq \emptyset$:
        $(A_1,B_1) = R.pop\_concept\_with\_min\_no\_attributes()$
        for $(A_2,B_2) \in \mathfrak{B}$ with $(A_1,B_1) \prec (A_2,B_2)$:
            $\tilde{s}_{(A_2,B_2)} = s_{(A_1,B_1)} + |(A_2 \times B_2) \setminus E|$
            if $\tilde{s}_{(A_2,B_2)} > s_{(A_2,B_2)}$:
                $s_{(A_2,B_2)} = \tilde{s}_{(A_2,B_2)}$
                $b_{(A_2,B_2)} = (A_1,B_1)$
                $R = R \cup \{(A_2,B_2)\}$
    $F_i := \emptyset$
    $b = \max(\mathfrak{B}\prec)$
    while $b \neq \perp$:
        $F_i = F_i \cup \{(A,B)\}$
        $b = b_{(A,B)}$
    $R = \{(g'',g') \mid g \in B, (A,B) \in F_i\}$
    $i = i+1$
return $F_1,\ldots,F_i$
```

the concepts $B_1 \ldots B_r$ are subsets of g'. We say that the object g *supports the factor* F_i until position r. If $r = j$ the object g *supports the whole factor F_i*.

For two objects $g_1, g_2 \in G$ in a dataset, we can now define the *ordinal factorization distance* as $d(g_1,g_2) = |g_1' \setminus g_2'|$. This distance between two objects counts the number of attributes that one object has, and the other one is missing. Note, that this distance is not symmetric and if $g_1' \subseteq g_2'$, it becomes 0. The Hamming distance, which is the symmetric version of the distance defined above, is commonly used to compute distances in binary datasets and can then be defined by

$$d_h(g_1,g_2) = d(g_1,g_2) + d(g_2,g_1).$$

We can generalize the notion of the ordinal factorization distance to positions in the factors. To do so, we compute the number of attributes in a factorization based on these positions. Then, the number of missing attributes of the object is computed. Let thus (G,M,I) be a formal context with a factorization $F_1,\ldots,F_k$ and $p_1,\ldots,p_k$ integers representing the position in each factor. Define the *ordinal factorization distance of the object g to the positions $p_1,\ldots,p_k$* as

$$d_F(g,p_1,\ldots,p_k) = |\{m \mid m \in F_i \text{ before position } p_i\ \} \setminus g'|.$$

The distance counts how many attributes an object is missing to support all factors F_i until position p_i. Note, that it is possible to which the *ordinal factorization distance* of an object is 0. This is trivially true, if for $p_1 = \cdots = p_k = 0$.

5.4.4 Ordinal Factorization to Discover Structure

We see the ordinal factor analysis as a tool to explain and discover relationships in unordered, bipartite datasets. The computationally expensive task to compute the factorization only has to be done once in the preprocessing step. To make the tool easily accessible, we envision a web platform that only has to do the relatively light computation of positions of objects in the ordinal factors. A user can then interactively navigate the dataset to discover the preexisting relationships in the data. To do so, we propose a navigation system consisting of slide controls. Each slide control thereby corresponds to one factor. In the case of an ordinal factorization with k factors, there are k slide controls. For each concept of a factor, there is a position in the corresponding slide control. As the top and bottom concepts of a factor usually have an empty object or attribute set, they do not get a corresponding position in the slide control. Additionally, a 0-position is introduced for each slide control to indicate an empty selection. Each position is annotated with the attributes that the corresponding concept gains compared to the previous one in the factor. An analyst can use the sliders to select the attribute positions for the slide controls. The system then displays the objects ordered by their ordinal factorization distance to the selected position. The second feature of this platform is that the analyst can select an object by clicking it. Then, the system automatically moves the slider for each factor to the maximal position the selected object supports. The remaining objects are then sorted based on the ordinal factorization distance of the selected position. For a prototype of how such a navigation system might work, we refer the reader to our demonstration platform[39]. We computed the ordinal factorizations of this tool using the greedy factor analysis as proposed in this work. The source code of this platform may be the basis for a more sophisticated exploration tool in the future.

5.5 Evaluation and Discussion

In this section we evaluate and discuss the usefulness of the two approaches that we explored in this chapter. As the usefulness of the two-dimensional projections of ordinal factor analysis for small datasets was explored before [58], we mainly discuss the runtime aspects of the two-dimensional part. For the greedy discovery of large factors, we perform a case study.

5.5.1 Runtime Discussion for Ordinal Two-Factorizations

If a formal context has order dimension two, it cannot contain a contranominal of dimension three as an induced subcontext. From a result by Albano [4], it follows that a context without a contranominal scale of dimension three and thus especially for all two-dimensional formal contexts, the number of concepts is bounded from above by $\frac{3}{2}|G|^2$ or dually $\frac{3}{2}|M|^2$. There are algorithms that compute the set of all concept of a formal context with polynomial delay [83] and the computation of a conjugate order can be performed in quadratic time [89]. Thus, the algorithm to compute ordinal two-factorizations has polynomial runtime if it paired with the these algorithms.

For the computation of large ordinal factorizations of formal contexts that do not omit ordinal two-factorizations, the runtime-obstacle is the computation of the large induced bipartite subgraph. If the exact problem is solved using the SAT-solver approach from Chapter 3, the algorithm has exponential runtime. We also discussed three heuristics for the computation of bipartite subgraphs in Chapter 3 which can be plugged to achieve an algorithm that has polynomial runtime. Usually, we are interested in two-factorizations of formal context with limited size, as a human can otherwise not grasp the connections encoded in the dataset. Thus, the runtime of these algorithms are usually not the critical limitation and thus a method that is computationally expensive can be employed.

5.5.2 Case Study of Greedy Ordinal Factorizations

Now, we demonstrate for five exemplary datasets that ordinal factor analysis is a method that can be applied to discover relationships in a dataset.

Source Code

Our algorithms are implemented in Python 3. An improved version of Lindigs algorithm [86, 80] is used to compute the covering relation of the concept lattice which we require. Even though we carefully implemented the code of our algorithms, it is focused on allowing experiments rather than speed. We therefore believe that it is possible that optimizing the code could result in a speed-up. The source code [40] is available for review and reproducibility.

Datasets

We conduct our experiments on multiple datasets from various sources, which we introduce in this section.

German AI and General AI. We generated this dataset with bibliometry data based on a work by Koopmann et.al. [76]. It contains the publication relationship between authors and conferences in the realm of artificial intelligence. The conferences in this dataset are based on an article by Kersting [75]. We generate two different datasets based on this work. The first is the AI community restricted to authors with a German affiliation, while the other is one contains all authors. We refer to the two datasets as *German AI* and *General AI*.

IMDb. We generated this dataset from the IMDb, a database that contains information on movies and series. They constitute the objects while their categories give rise to the attributes. We use the data as it was on the 6th of October 2021 [69] and refer to this dataset as the *IMDb* dataset.

Wahl-O-Mat. The Wahl-O-Mat is a website provided by the “Bundeszentrale für politische Bildung” [15] in Germany. It is a tool that helps german citizens decide which parties to vote for in federal elections. Thereby, a set of statements is provided on which the citizens have to decide whether they agree or disagree. Afterwards, the system compares the results to statements of different parties and suggests which party might be a good fit for the views of the specific citizen. We

Table 5.1: Descriptive measures for the datasets used in the case study.

	Ger-AI	AI	IMDb	parties	statem.
Attributes	133	137	28	38	76
Objects	2238	218308	3784644	76	38
Concepts	12709	1981691	6765616	24245	24245
Density	0.0258	0.0149	0.0501	0.429	0.429
Mean attr. per concept	4.87	7.24	5.64	7.26	8.28
Mean obj. per concept	7.35	13.54	15.76	8.28	7.26
Covering relation size	46776	11455411	284543420	24245	24245
Greedy ordinal factors	129	137	28	20	20

Table 5.2: Experimental runtime of the algorithm. The computation of the concepts has to be performed for both algorithms, i.e., for the naive approach and for OrdiFIND

	Concepts	Naive	OrdiFIND
General AI	4.63h	4.18h	3.57h
German AI	3.76s	33.54s	12.97s
IMDb	4.46d	>30d	14.42d
Wahl-O-Mat parties	4.34s	16.92s	11.96s
Wahl-O-Mat statements	8.18s	14.23s	13.36s

extracted the statements and parties of the 2021 federal election to create two datasets. The first one, which we refer to as *Wahl-O-Mat-parties* has the parties as objects and the statements as attributes. The dataset *Wahl-O-Mat-statements* has the statements as objects and the parties as attributes.

Runtime

We performed all computations on an Intel Xeon Gold 5122 CPU equipped with 800 GB of memory. For each dataset, we used the methods from this work to compute a complete ordinal factorization. In Table 5.2, the runtime of OrdiFIND is compared to the naive algorithm. Also, the runtime of the computation of the concept lattice is listed. One can observe that in almost all instances, the computation of the covering relation takes a substantial amount of time. It is also clearly visible that OrdiFIND outperforms the naive version of the algorithm. We were not able to compute the factorization of the IMDb dataset in less than 30 days using the naive algorithm. All in all, we conclude that it is feasible to apply the method proposed in this paper on all datasets, for which it is still possible to compute the concept lattice.

Interesting Findings

In this section, we demonstrate a few examples on how to use the method proposed in this chapter to discover interesting relationships in binary data and discuss findings in our datasets.

German AI and General AI. The greedy factorization of the German AI and the General AI datasets results in 129 and 137 factors, respectively. For the German AI dataset, we include the factors in Figure 5.9, the prototype web platform from Section 5.4.4 provides insight for the General AI community. It seems sensible

1. ICRA > IROS > I. J. Robotics Res. > IEEE Trans. Robotics > Robotics and Autonomous Systems > Auton. Robots > IEEE Robotics and Automation Letters > IEEE Robot. Automat. Mag. > J. Field Robotics > Robotics: Science and Systems > FSR > ISRR > AAAI > IJCAI > NIPS, ICML > Humanoids, Artif. Intell., J. Artif. Intell. Res., Machine Learning > ACL, AAMAS, CoRL, IEEE Trans. Pattern Anal. Mach. Intell., ICCV, UAI, J. Mach. Learn. Res., ICAPS
2. CVPR > ICCV > ECCV > IEEE Trans. Pattern Anal. Mach. Intell. > International Journal of Computer Vision > Computer Vision and Image Understanding > ACCV > AAAI > IJCAI > NIPS > ICML > IEEE Trans. Neural Networks > NeurIPS > IJCNN > IEEE Trans. Knowl. Data Eng., ICDM > SDM, KDD, AISTATS > IEEE Trans. Neural Netw. Learning Syst., IEEE Trans. Systems, Man, and Cybernetics, Part C, Neural Computation > Data Min. Knowl. Discov., NAACL-HLT, Machine Learning, IROS, UAI, J. Mach. Learn. Res.
3. AAAI > IJCAI > CIKM > KDD > ICDM > IEEE Trans. Knowl. Data Eng. > SDM > PAKDD > WWW > WSDM > SIGIR > ECML/PKDD > ICML > ACL > NIPS > CVPR > UAI, IEEE Trans. Pattern Anal. Mach. Intell. > Artif. Intell. > Machine Learning > ACL/IJCNLP, International Semantic Web Conference, EMNLP/IJCNLP, ECML > NeurIPS, HLT-NAACL, ICRA, CoNLL, Autonomous Agents and Multi-Agent Systems, NAACL-HLT, PKDD, ML, Robotics and Autonomous Systems, ECAI, COLT, AISTATS, J. Mach. Learn. Res., EMNLP, IROS, COLING, EMNLP-CoNLL
4. IJCNN > Neural Networks > IEEE Trans. Neural Networks > ICANN > Neural Computation > NIPS > J. Mach. Learn. Res. > IEEE Trans. Pattern Anal. Mach. Intell. > Machine Learning > ICML > ECML/PKDD > UAI > SDM > KDD > COLT > EMNLP/IJCNLP, AAAI, PAKDD, AISTATS, RecSys > HLT-NAACL, CVPR, International Journal of Computer Vision, CIKM, SIGIR, WSDM, EACL, ICCV, ECML, WWW, EMNLP-CoNLL
5. IROS > Robotics and Autonomous Systems > Auton. Robots > IEEE Robot. Automat. Mag. > ICRA > Humanoids > ISRR > Robotics: Science and Systems > IEEE Robotics and Automation Letters > I. J. Robotics Res. > NIPS > IJCAI > AISTATS > J. Mach. Learn. Res. > ICML, CoRL, ECML, IEEE Trans. Robotics, Neural Networks, Neural Computation, IJCNN > ECML/PKDD, IEEE Trans. Neural Netw. Learning Syst., Artif. Intell., AAAI, ICANN, AAMAS, Machine Learning, ECAI, IEEE Trans. Pattern Anal. Mach. Intell., ICAPS
6. NIPS > ICML > AISTATS > J. Mach. Learn. Res. > UAI > Machine Learning > COLT > NeurIPS > KDD > ICDM > IJCAI > IEEE Trans. Pattern Anal. Mach. Intell. > AAAI > Neural Computation > ICRA, IEEE Trans. Neural Networks, SDM, Neural Networks, IJCNN > Humanoids, ICANN, Auton. Robots, PAKDD, IROS > Artif. Intell., IEEE Robotics and Automation Letters, CVPR, International Journal of Computer Vision, ACCV, PKDD, I. J. Robotics Res., WSDM, ECCV, ICCV, ECML, WWW, Robotics: Science and Systems, ICLR
7. CHI > UbiComp > ACM Trans. Comput.-Hum. Interact. > IUI > IJCAI > AAMAS > J. Artif. Intell. Res. > HCOMP, Artif. Intell., AAAI > WWW, UAI > HLT-NAACL, ACL, ICML, CIKM, KDD, WSDM, EMNLP > ACL/IJCNLP, IEEE Trans. Knowl. Data Eng., KR, EMNLP/IJCNLP, NAACL-HLT, Machine Learning, AIPS, ICAPS
8. IJCAI > Artif. Intell. > ECAI > J. Artif. Intell. Res. > KR > JELIA > TPLP > ICLP > AAAI > LPAR > RR > ESWC > IEEE Trans. Knowl. Data Eng., International Semantic Web Conference, RuleML > RuleML+RR > IJCAR, ILP, CPAIOR, CP, WWW, TARK
9. CIKM > SIGIR > WWW > WSDM > KDD > ACL > EMNLP > EMNLP/IJCNLP > AAAI > NAACL-HLT > NeurIPS > NIPS, ICML > IJCAI > ECML/PKDD > J. Artif. Intell. Res. > COLT, ICDM, Machine Learning > ACL/IJCNLP, HLT-NAACL, Artif. Intell., HLT/EMNLP, ML, RecSys, SDM, UAI, EMNLP-CoNLL, ICLR > CoNLL, PAKDD, EACL, ICCV, COLING, J. Mach. Learn. Res.
10. ACL > EMNLP > COLING > HLT-NAACL > EACL > NAACL-HLT > EMNLP/IJCNLP > CoNLL > EMNLP-CoNLL > HLT/EMNLP > J. Artif. Intell. Res. > AAAI > ICML > IJCAI, KDD, Machine Learning, NIPS, J. Mach. Learn. Res. > COLING-ACL > IEEE Trans. Knowl. Data Eng., KR, ILP, AAMAS, PKDD, ECCV, WWW, ECML/PKDD, CVPR, CIKM, IEEE Trans. Pattern Anal. Mach. Intell., NeurIPS, Artif. Intell., ECML, Neural Computation, COLT, AISTATS, ICDM, J. Autom. Reasoning

Figure 5.9: The attributes of the first 10 ordinal factors from the German AI dataset. It is notable that the factors cluster conferences with a similar focus into the same factors.

to interpret the single factors as communities of conferences, where authors of similar interest publish. For example, the first factor in both cases consists of Robotics conferences. The second factor of the German AI dataset corresponds then to the computer vision community, while the third factor contains the general AI conferences. Because of the nature of ordinal factorizations, the conferences that appear early in such a factor are the more general ones. The ones that are deep into a factor are more specialized. An ordinal factor analysis can thus be leveraged by an author as a tool to select a conference.

It is interesting to note that the communities discovered with this method mostly but not fully coincide with the AI communities formulated by Kersting [75].

1. Comedy > Drama > Romance > Family > Crime > Talk-Show > News > Music > Game-Show > Reality-TV > Documentary > Sport > Adventure > Action > Animation > Mystery > Musical > Fantasy > Horror > Thriller > Biography > History > War > Short > Sci-Fi > Western > Adult > Film-Noir
2. Short > Documentary > Drama > Action > Adventure > Animation > Crime > Mystery > Thriller > Horror > Fantasy > Sci-Fi > Reality-TV > Family > Music > History > Comedy > Biography > Adult > Romance > Sport > Western > War > Musical > Game-Show > News > Talk-Show > Film-Noir
3. Drama > Crime > Mystery > Romance > News > Talk-Show > Family > Comedy > Adventure > Animation > Fantasy > Documentary > Action > Musical > Reality-TV > Horror > Sport > History > Biography > Music > Sci-Fi > War > Thriller > Short > Western > Game-Show > Adult > Film-Noir
4. Documentary > News > Talk-Show > Game-Show > Family > Reality-TV > Music > Sport > Action > Adventure > Animation > Comedy > History > Biography > Drama > Sci-Fi > Mystery > Horror > Romance > Thriller > Short > Crime > Western > War > Fantasy > Musical > Adult > Film-Noir
5. Animation > Adventure > Action > Family > Fantasy > Horror > Thriller > Sci-Fi > Drama > Mystery > Adult > Biography > Comedy > History > Documentary > War > Romance > Short > Western > Sport > Music > Reality-TV > Crime > Film-Noir > Game-Show > Talk-Show > News, Musical

Figure 5.10: The attributes of the first five ordinal factors of a greedy ordinal factorization for the IMDb dataset.

IMDb. The complete greedy factorization of the movie dataset consists of 27 factors, the first 10 are depicted in Figure 5.10. Each factor corresponds to a linear order of genres. One can derive knowledge about the movie landscape using these factors as follows. The first factor seems to focus on light entertainment. As most movie productions are in this factor, this is one with the highest relevance for the industry. The second factor then has a more educational focus. The third one is once again a factor with entertaining shows, however, the focus in this factor seems to be less light than in the first one. This factorization is valuable, as it allows to navigate the movie database.

We furthermore believe that it is possible to extend the current method to a more sophisticated recommendation paradigm based on these factors. The users could then select, for example, that they want to view a movie that is highly entertaining but also a bit educational. Then the system would recommend movies based on the ordinal factor distance.

Wahl-O-Mat. We use two different interpretations of the Wahl-O-Mat dataset for our research. The reason is that, in our opinion, both datasets are of interest for examination with ordinal factor analysis. In the first one, the attributes (and thus the factors) contain the political parties, while in the second case, they contain statements. When discussing political topics, it is common to refer to statements as progressive or conservative. Often, these statements are even compared with each other concerning their progressiveness or conservativeness. Thus, the everyday intercourse with political topics implies that there is a linear order on the positions. The same is true for political parties, which are commonly also ordered similarly. However, in a diverse political landscape, a single order is most likely not adequate to analyze the political viewpoints. In our dataset, we are able to identify that some parties which are typically classified as right-wing also advocate some progressive positions such as raising the minimum wage. The ordinal factor analysis enables a more sophisticated investigation of this situation. Each factor provides a linear order which arranges the parties or statements respectively. Thereby, each factor is consistent with the positions of a party. If we analyze the statement dataset, we can observe that the classical progressive positions are condensed in the first factor, while the second factor contains the typical conservative ones. The other factors are more specialized, as the order of the positions bases not on the conservative-progressive spectrum but, for example, migration or economics. In every factor, the early statements are the once where many parties have a consensus. Thus, they are the ones that have a high probability to be implemented by a future government.

Limitations

The main limitation of the method is the computational feasibility. The algorithm has to solve an NP-complete problem which it does in an exponential-time algorithm with respect to the size of the input data. Thus, it is not feasible for large datasets, where it is not currently possible to compute the concept lattice.

5.6 Conclusion

In this section, we followed two lines of research in the realm of ordinal factor analysis. First, we investigated the computation of maximal ordinal two-factorizations. To this end, we develped a polynomial time algorithm to compute a two-factorization of a formal context that has a bipartite incompatibility graph. We showed, that the problem to compute maximal ordinal two-factorizations is NP-complete and proposed our approch ORD2FACTOR to compute large ordinal two-factorizations. Then, we proposed the algorithm ORDIFIND by leveraging fast algorithms developed in formal concept analysis. The resulting algorithm enabled us to compute a greedy ordinal factorization for larger unordered binary datasets. Building on these factorizations, we demonstrated how to discover relationships in the original data. Both lines of research are linked to NP-complete problems and both resulting algorithms run in exponential time with respect to the input data.

Datasets often consist not only of binary but also already ordinal data. The ordinal factor analysis in its current form can only deal with this data by interpreting it as binary. While scaling in formal concept analysis is a tool to deal with this data, factors will not necessarily respect the order encapsulated in the data. In our opinion, the next step should be to extend this method to deal with this kind of non-binary data directly.

Part III

Visualization of Concept Lattices

CHAPTER 6

Force Directed Order Diagram Drawing

Order diagrams allow human analysts to understand and analyze structural properties of ordered data. While an experienced human can create easily readable diagrams, the automatic generation of those remains a hard task. This is closely related to the graph drawing problem, which poses the research question:

(2.1) How can we adapt graph drawing techniques to order diagram drawing?

Force-directed approaches are widely applied, and it is generally accepted that they generate aesthetically-pleasing drawings of graphs. In this chapter, we adapt this method to the realm of order diagram drawing. Our algorithm ReDraw thereby embeds the order into a high dimensional Euclidean space and then iteratively reduces the dimension until a two-dimensional drawing is achieved. To attain aesthetical pleasing results, between each dimension reduction step two force-directed steps are performed. Those optimize the distances of nodes and the distances of lines in order to satisfy a set of a priori fixed conditions. By respecting an invariant about the vertical position of the elements in each step, our algorithm ensures that the resulting drawings satisfy all necessary properties of order diagrams.

6.1 Introduction

The availability of visualizations of ordered sets is an integral requirement for developing and applying ordinal data science methods such as formal concept analysis. They are a necessity to turn semi-automated data exploration into a mature instrument. Thus, the automatic generation of order diagrams is an important research line.

The general structure of an order diagram is dictated by a set of *hard constraints* which stem from the requirement for the diagram to precisely represent the comparabilities of the ordered set. Every element is visualized by a node and two elements are connected by a straight line if and only if one is lesser than the other and there is no element "in between". Moreover, it has to hold for comparable elements that the node representing the greater element has a larger y-coordinate than the lesser node. Furthermore, no two nodes are allowed to be positioned on the same coordinates. Finally, nodes are not allowed to touch non-adjacent lines. For an exact definition of an order diagram, see Definition 2.28.

While these requirements give direction on how the order diagram should look like, they are not sufficient for generating drawings that are perceived as "readable" by humans. On the contrary, finding good coordinates that result in aesthetically pleasing results for the nodes is very challenging. The creation of an order diagram relies on respecting the hard criteria and trying to satisfy a set of commonly accepted but frequently conflicting *soft criteria*. Those are generally considered to make a drawing more readable [120]. They include maximizing the distances between element nodes and lines, minimizing the number of crossing lines, maximizing the angles of crossing lines, minimizing the number of different line directions or organizing the nodes in a limited number of layers. Experts with enough practice can create such drawings; however, this is a time-consuming and thus uneconomical task and therefore rather uncommon. They thereby balance the criteria based on their own perception. However, it is unclear how to transfer this balancing act to an algorithm. Even if an algorithm that optimizes on these criteria would exist, it might not even yield readable results as prior works [24] suggests. Another obstacle on this way is that not every human reader perceives the same aspects of an order diagram as "readable" but quite conversely different individuals perceive different aspects of a good drawing as important. It is thus hardly possible to develop a good fitness function for readable graphs. Those reasons combined make the automatic generation of readable graph drawing - and even their evaluation - a surprisingly hard task. None of the previously proposed algorithms produce drawings that are able to compete with the drawings that are manually drawn by an expert.

In this section, we try to address this problem by proposing our new algorithm ReDraw that adapts the force-directed approach of graph drawing to the realm of order diagram drawing. Thereby, a physical simulation is performed in order to optimize a drawing by moving it to a state of minimal stress. The final results of the drawings computed by our algorithm are sufficiently readable. We compare the drawings generated by our approach to prior algorithms and show they are more readable under certain conditions.

6.2 Related Work

Order diagram drawing can be considered to be a special version of the graph drawing problem, where a graph is given as a set of vertices and a set of edges and a readable drawing of this graph is desired. Thereby, each vertex is once again represented by a node and two adjacent vertices are connected by a straight line. The graph drawing problem suffers from a lot of the same challenges as order diagram drawing. For a graph, it can be checked in linear time whether it is planar [68]), i.e., whether it has a drawing that has no crossing edges. In this case a drawing only consisting of straight lines without crossings, bends, and curves can always be computed [94] and should thus be preferred. For a directed graph with a unique maximum and minimum, like for example a lattice, it can be checked in linear time whether an upward planar drawing exists. Then, such a drawing can be computed in linear time [8]. The work of Battista et al. [8] provides an algorithm to compute straight-line drawings for "serial parallel graphs", which is a special family of planar, acyclic graphs. As symmetries are often preferred by readers, the algorithm was extended [67] to reflect them in the drawings based on the automorphism group of the graph. However, lattices that are derived from real world data rarely satisfy the planarity property [3].

The most successful approaches for order diagram drawing are a work of Sugiyama et al. [104] which is usually referred to as Sugiyama's framework and a work of Freese [50]. Both algorithms use the structure of the ordered set to decide on the height of the element nodes; however, the approach choosing the horizontal coordinates of a node differ significantly. While Sugiyama's framework minimizes the number of crossing lines between different vertical layers, Freese's layout adapts a force-directed algorithm to compute a three-dimensional drawing of the ordered set. Another force-directed approach that is based on minimizing a "conflict distance" is suggested in [123].

In this chapter, we propose the force-directed graph drawing algorithm ReDraw that, similarly to Freese's approach, operates not only in two but in higher dimensions. Compared to Freese's layout, our algorithm however starts in an arbitrarily high dimension and improves it then by reducing the number of dimensions in an iterative process. Thus, it minimizes the probability to stop the algorithm early with a less pleasing drawing. Furthermore, our approach gets rid of the ranking function to determine the vertical position of the elements and instead uses the force-directed approach for the vertical position of nodes as well. We achieve this by defining a vertical invariant which is respected in each step of the algorithm. This invariant guarantees that the resulting drawing will respect the hard condition of placing greater elements higher than lesser elements.

6.2.1 Force-Directed Approaches

We now provide some additional notations which are not necessarily standard but will be used throughout this chapter. We start with a generalization of the order diagram notion tho arbitrary high dimensions.

Definition 6.1 (D-Dimensional Order Diagram)
A d-dimensional *order diagram* or *drawing* of an ordered set $(X, \leq)$ is denoted by $(\vec{p}_a)_{a \in X} \subseteq \mathbb{R}^d$ whereby $\vec{p}_a = (x_{a,1}, \ldots, x_{a,d-1}, y_a)$ for each $a \in X$ and for all $a \prec b$ it holds that $y_a < y_b$.

If the dimension of an order diagram is not further qualified, the two-dimensional case is assumed. To fully specify the drawing, one additionally has to describe the lines connecting the elements in covering relation. We consider all lines, as straight in this work. By the definition above the last element of the vector provides the structural comparabilities of the ordered set. To further specify this, we call the parts and forces as follows.

Definition 6.2 (Horizontal and Vertical Components and Forces)
We call y_a the *vertical component* and $x_{a,1}, \ldots, x_{a,d-1}$ the *horizontal components* of $\vec{p}_a$ and denote $(\vec{p}_a)_x = (x_{a,1}, \ldots, x_{a,d-1}, 0)$. The forces operating on the vertical component are called the *vertical force* and the forces operating on the horizontal components the *horizontal forces*.

Our algorithm optimizes the horizontal and vertical positions of the order diagram independently. We thus define those distance, such that we can refer to them separately.

Definition 6.3 (Horizontal and Vertical Distances)
The Euclidean distance between the representation of a and b is denoted by

$$d(\vec{p}_a, \vec{p}_b) = |\vec{v}_a - \vec{v}_b|,$$

while the distance between the vertical components is denoted by

$$d_y(\vec{p}_a, \vec{p}_b)$$

and the distance in the horizontal components is denoted by

$$d_x(\vec{p}_a, \vec{p}_b) = d((\vec{p}_a)_x, (\vec{p}_b)_x).$$

For a more compact notation, we specify the vectors in a direction of length one instead of normalizing them in each step.

Definition 6.4 (Unit Vectors)
The unit vector from $\vec{p}_a$ to $\vec{p}_b$ is denoted by $\vec{u}(\vec{p}_a, \vec{p}_b)$, the unit vector operating in the horizontal dimensions is denoted by $\vec{u}_x(\vec{p}_a, \vec{p}_b)$.

The cosine distance allows to investigate the angle between two lines.

Definition 6.5 (Cosine Distance)
Finally, the *cosine-distance* between two vector pairs $(\vec{a}, \vec{b})$ and $(\vec{c}, \vec{d})$ with $\vec{a}, \vec{b}, \vec{c}, \vec{d} \in \mathbb{R}^d$ is given by

$$d_{\cos}((\vec{a}, b), (\vec{c}, \vec{d})) := 1 - \frac{\sum_{i=1}^{d}(b_i - a_i) \cdot (d_i - c_i)}{d(a,b) \cdot d(c,d)}.$$

The general idea of force-directed algorithms is to represent the graph as a physical model consisting of steel rings each representing a vertex. For every pair of adjacent vertices, their respective rings are connected by identical springs, i.e., springs with identical length and spring constant. Using a physical simulation, this system is then moved into a state of minimal stress, which can in turn be used as the drawing. Many modifications to this general approach, that are not necessarily based on springs, were proposed in order to encourage additional conditions for the resulting drawings.

Algorithm 6.11 Force-Directed Graph Drawing Algorithm by Eades

Input: Graph (V,E)
Initial drawing $p = (\vec{p}_a)_{a\in V} \subseteq \mathbb{R}^2$
Maximum number of iterations $K \in \mathbb{N}$
Minimal stress $\varepsilon > 0$
Damping factor $\delta > 0$
Output: Coordinates for the drawing $p = (\vec{p}_a)_{a\in V} \subseteq \mathbb{R}^2$

```
def force_directed($V,E,p,K,\varepsilon,\delta$):
    $t$ = 1
    while $t < K$ and $\max_{a\in V} \|F_a(t)\| > \varepsilon$:
        for $a \in V$:
            $F_a(t)$ = $\sum_{\{a,b\}\notin E} f_{\text{rep}}(\vec{p}_a,\vec{p}_b) + \sum_{\{a,b\}\in E} f_{\text{spring}}(\vec{p}_a,\vec{p}_b)$
        for $a \in V$:
            $\vec{p}_a$ = $\vec{p}_a + \delta \cdot F_a(t)$
        $t$ = $t+1$
```

The idea of force-directed algorithms was first suggested by Eades [43]. His algorithmic realization of this principle uses an iterative approach where in each step of the simulation the forces that operate on each vertex are computed and summed up (cf. Algorithm 6.11). Based on the sum of the forces operating on each vertex, they are then moved. This is repeated for either a limited number of rounds or until there is no stress left in the physical model. While a system consisting of realistic springs would result in linear forces between the vertices, Eades claims that those are performing poorly and thus introduces an artificial spring force. This force operates on each vertex a for adjacent pairs $\{a,b\} \in E$ and is given as

$$f_{\text{spring}}(\vec{p}_a,\vec{p}_b) = -c_{\text{spring}} \cdot \log\left(\frac{d(\vec{p}_a,\vec{p}_b)}{l}\right) \cdot \vec{u}(\vec{p}_a,\vec{p}_b),$$

whereby c_{spring} is the spring constant and l is the equilibrium length of the spring. The spring force repels two vertices if they are closer then this optimal distance l while it operates as an attracting force if two vertices have a distance greater then l, see Figure 6.1. To enforce that non-connected vertices are not placed too close to each other, he additionally introduces the repelling force that operates between non-adjacent vertex pairs as

$$f_{\text{rep}}(\vec{p}_a,\vec{p}_b) = \frac{c_{\text{rep}}}{d(\vec{p}_a,\vec{p}_b)^2} \cdot \vec{u}(\vec{p}_a,\vec{p}_b).$$

The value for c_{rep} is once again constant. In a realistic system, even a slightest movement of a vertex changes the forces that are applied to its respective ring. To depict this realistically a damping factor δ is introduced in order to approximate the realistic system. The smaller this damping factor is chosen, the closer the system is to a real physical system. However, a smaller damping factor results in higher computational costs. In some instances this damping factor is replaced by a cooling function $\delta(t)$ to guarantee convergence. The physical simulation stops if the total stress of the system falls below a constant ε. Building on this approach, a modification is proposed in the work of Fruchterman and Reingold [51] from 1991. In their algorithm, the force

$$f_{\text{attr}}(\vec{p}_a, \vec{p}_b) = -\frac{d(\vec{p}_a, \vec{p}_b)^2}{l} \cdot \vec{u}(\vec{p}_a, \vec{p}_b)$$

is operating between every pair of connected vertices. Compared to the spring-force in Eades' approach, this force is always an attracting force. Additionally, the force

$$f_{\text{rep}}(\vec{p}_a, \vec{p}_b) = \frac{l^2}{d(\vec{p}_a, \vec{p}_b)} \cdot \vec{u}(\vec{p}_a, \vec{p}_b)$$

repels every vertex pair. Thus, the resulting force that is operating on adjacent vertices is given by

$$f_{\text{spring}}(\vec{p}_a, \vec{p}_b) = f_{\text{attr}}(\vec{p}_a, \vec{p}_b) + f_{\text{rep}}(\vec{p}_a, \vec{p}_b)$$

and has once again its equilibrium at length l. These forces are commonly considered achieving better drawings than Eades' approach and are thus usually preferred.

While the graph drawing algorithms described above lead to sufficient results for undirected graphs, they are not suited for order diagram drawings, as they do not take the direction of an edge into consideration. Therefore, they will not satisfy the hard condition that greater elements have higher y-coordinates. Freese [50] proposed an algorithm for lattice drawing that operates in three dimensions, where the ranking function

$$rank(a) = height(a) - depth(a)$$

fixes the vertical position of each element. The function $height(a)$ thereby evaluates to the length of the longest chain between a and the minimal element and the function $depth(a)$ to the length of the longest chain to the maximal element. While this ranking function guarantees that lesser elements are always positioned below greater elements, the horizontal coordinates are computed using a force-directed

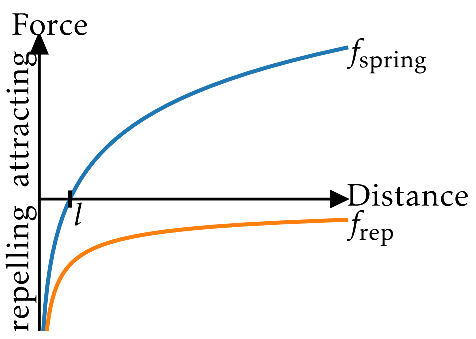


Figure 6.1: The forces for graphs as introduced by Eades in 1984. The f_{spring} force operates between adjacent vertices and has an equilibrium at l, the repelling force f_{rep} operates on non-adjacent pairs.

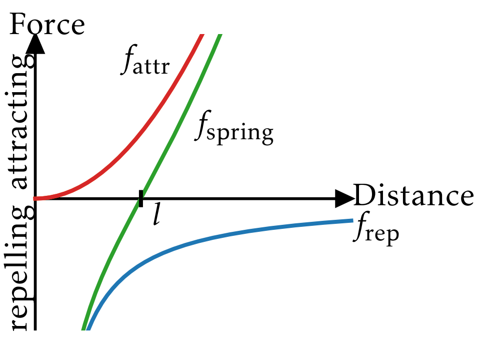


Figure 6.2: The forces as introduced by Fruchterman and Reingold. The f_{rep} force operates between all vertex pairs while adjacent vertices attract each other with f_{attr}. The resulting force on adjacent vertices is f_{spring}.

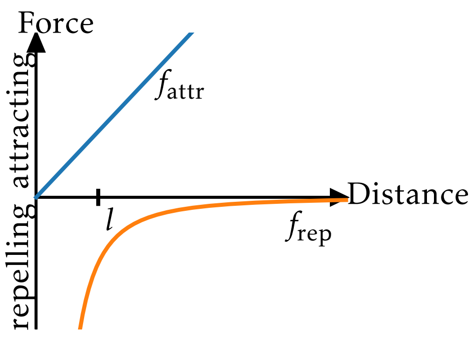


Figure 6.3: Horizontal forces for drawing order diagrams introduced by Freese in 2004. The force f_{attr} operates between comparable pairs, the force f_{rep} between incomparable pairs. There is no vertical force.

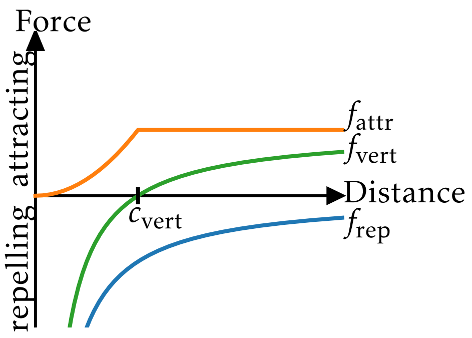


Figure 6.4: Our forces for drawing order diagrams. f_{vert} operates vertically between node pairs in the covering relation, the force f_{attr} between comparable pairs and the force f_{rep} between incomparable pairs.

approach. Freese introduces an attracting force between comparable elements that is given by

$$f_{\text{attr}}(\vec{p}_a, \vec{p}_b) = -c_{\text{attr}} \cdot d_x(\vec{p}_a, \vec{p}_b) \cdot \vec{u}_x(\vec{p}_a, \vec{p}_b),$$

and a repelling force that is given by

$$f_{\text{rep}}(\vec{p}_a, \vec{p}_b) = c_{\text{rep}} \cdot \frac{d_x(\vec{p}_a, \vec{p}_b)}{|y_b - y_a|^3 + |x_{b,1} - x_{a,1}|^3 + |x_{b,2} - x_{a,2}|^3} \cdot \vec{u}_x(\vec{p}_a, \vec{p}_b)$$

operating on incomparable pairs only, (cf. Figure 6.3). The values for c_{attr} and c_{rep} are constants. A parallel projection is either done by hand or chosen automatically to compute a two-dimensional depiction of the three-dimensional drawing.

6.3 The ReDraw Algorithm

Our algorithm ReDraw uses a force-directed approach similar to the one that is used in Freese's approach. Compared to Freese's algorithm, we however do not use a static ranking function to compute the vertical positions in the drawing. Instead, we use forces which allow us to incorporate additional properties like the horizontal distance of vertex pairs into the vertical distance. By respecting a vertical invariant, that we will describe later, the vertical movement of the vertices is restricted so that the hard constraint on the y-coordinates of comparable nodes can be always guaranteed. However, the algorithm is thus more likely to get stuck in a local minimum. We address this problem by computing the first drawing in a high dimension and then iteratively reducing the dimension of this drawing until a two-dimensional drawing is achieved. This is done, as the additional degrees of freedom will allow the drawing to move less restricted in higher dimensions. Thus, the probability for the system to get stuck in a local minimum is reduced. Our algorithm framework (cf. Algorithm 6.12) consists of three individual algorithmic steps that are iteratively repeated. We call one repetition of all three steps a *cycle*. In each cycle, the algorithm is initialized with the d-dimensional drawing and returns a $(d-1)$-dimensional drawing. The first step of the cycle, which we refer to as the *node step*, improves the d-dimensional drawing by optimizing the proximity of nodes in order to achieve a better representation of the ordered set. In the second step, which we call the *line step*, the force-directed approach is applied to improve distances between different lines as well as between lines and nodes. The resulting drawing thereby achieves a better satisfaction of soft criteria and

Algorithm 6.12 ReDraw Algorithm to Compute Force Directed Order Diagrams

Input: Ordered set $O = (X, \leq)$
Initial dimension d
Constants $K \in \mathbb{N}$, $\varepsilon > 0$, $\delta > 0$, $c_{\text{vert}} > 0$, $c_{\text{hor}} > 0$, $c_{\text{par}} > 0$, $c_{\text{ang}} > 0$, $c_{\text{dist}} > 0$
Output: Drawing: $p = (\vec{p}_a)_{a \in V} \subseteq \mathbb{R}^2$

```
def redraw($O, d, K, p, d, K, \delta, c_{\text{vert}}, c_{\text{hor}}, c_{\text{par}}, c_{\text{ang}}, c_{\text{dist}}$):
    $p$ = initial_drawing($O$)
    while $d \geq 2$:
    node_step($O, p, d, K, \varepsilon, \delta, c_{\text{vert}}, c_{\text{hor}}$)
    line_step($O, p, d, K, \varepsilon, \delta, c_{\text{par}}, c_{\text{ang}}, c_{\text{dist}}$)
    if $d > 2$:
        dimension_reduction($O, p, d$)
        $d$ = $d - 1$
```

thus improves the readability for a human reader. Finally, in the *reduction step* the dimension of the drawing is reduced to $(d-1)$ by using a parallel projection into a subspace that preserves the vertical dimension. In the last (two-dimensional) cycle, the dimension reduction step is omitted.

The initial drawing used in the first cycle is randomly generated. The vertical coordinate of each node is given by its position in a randomly chosen linear extension of the ordered set. The horizontal coordinates of each element are set to a random value between -1 and 1. This guarantees that the algorithm does not start in an unstable local minimum. Every further cycle then uses the output of the previous cycle as input to further enhance the resulting drawing.

Compared to the approach of Freese, we do not fix the vertical component by a ranking function. Instead, we recompute the vertical position of each element in each step using our force-directed approach. To ensure that the resulting drawing is in fact a drawing of the ordered set, we guarantee that in every step of the algorithm the following property is satisfied:

Definition 6.6 (Vertical Constraint)
Let $(X,\leq)$ be an ordered set with a drawing $(\vec{p}_a)_{a\in X}$. The drawing $(\vec{p}_a)_{a\in X}$ *satisfies the vertical constraint*, iff $\forall a,b \in X : a < b \Rightarrow y_a < y_b$.

This vertical invariant is preserved in each step of the algorithm and thus in the final drawing the comparabilities of the order are correctly depicted.

6.3.1 Node Step

The first step of the iteration is called the *node step*, which is used in order to compute a d-dimensional representation of the ordered set. It thereby emphasizes the ordinal structure by positioning element pairs in a similar horizontal position, if they are comparable. In this step, we define three different forces that operate simultaneously. For each $a \leq b$ on a the vertical force

$$f_{\text{vert}}(\vec{p}_a,\vec{p}_b) = \left(0,\ldots,0,-c_{\text{vert}}\cdot\left(\frac{1+d_x(\vec{p}_a,\vec{p}_b)}{d_y(\vec{p}_a,\vec{p}_b)}-1\right)\right)$$

operates while on b the force $-f_{\text{vert}}(\vec{p}_a,\vec{p}_b)$ operates. If two elements have the same horizontal coordinates, it has its equilibrium if the vertical distance is at the constant c_{vert}. Then, if two elements are closer than this constant, it operates repelling and if they are farther away, the force operates as an attracting force.

Algorithm 6.13 The Node Step of ReDraw

Input: Ordered set $(X, \leq)$
Drawing $p = (\vec{p}_a)_{a \in X} \subseteq \mathbb{R}^d$
Constants $K \in \mathbb{N}$, $\varepsilon > 0$, $\delta > 0$, $c_{\text{vert}} > 0$, $c_{\text{hor}} > 0$
Output: Drawing: $p = (\vec{p}_a)_{a \in X} \subseteq \mathbb{R}^d$

```
def node_step($O, p, d, K, \varepsilon, \delta, c_{\text{vert}}, c_{\text{hor}}$):
    $t = 1$
    while $t < K$ and $\max_{a \in X} \|F_a(t)\| > \varepsilon$:
    for $a \in X$:
        $F_a(t) = \sum_{a<b} f_{\text{vert}}(\vec{p}_a, \vec{p}_b) - \sum_{b<a} f_{\text{vert}}(\vec{p}_a, \vec{p}_b)$
                $+ \sum_{a \leq b} f_{\text{attr}}(\vec{p}_a, \vec{p}_b) + \sum_{a \not\leq b} f_{\text{rep}}(\vec{p}_a, \vec{p}_b)$
    for $a \in X$:
        $\vec{p}_a$ = overshooting_protection($\vec{p}_a + \delta \cdot F_a(t)$)
    $t = t + 1$
```

Thus, the constant c_{vert} is a parameter that can be used to tune the *optimal vertical distance*. By incorporating the horizontal distance into the force, it can be achieved that vertices with a high horizontal distance will also result in a higher vertical distance. Note, that this force only operates on the covering relation instead of all comparable pairs, as otherwise, chains would be contracted to be positioned close to a single point.

On the other hand, there are two different forces that operate in the horizontal direction. Similar to Freese's layout, there is an attracting force between comparable and a repelling force between incomparable element pairs; however, the exact forces are different. Between all comparable pairs a and b, the force

$$f_{\text{attr}}(\vec{p}_a, \vec{p}_b) = -\min\left(d_x(\vec{p}_a, \vec{p}_b)^3, c_{\text{hor}}\right) \cdot \vec{u}_x(\vec{p}_a, \vec{p}_b)$$

is operating. Note that, in contrast to f_{vert}, this force operates not only on the covering but on all comparable pairs and thus encourages chains to be drawn in a single line. Similarly, incomparable elements should not be close to each other and the force

$$f_{\text{rep}}(\vec{p}_a, \vec{p}_b) = \frac{c_{\text{hor}}}{d_x(\vec{p}_a, \vec{p}_b)} \cdot \vec{u}_x(\vec{p}_a, \vec{p}_b)$$

repels incomparable pairs horizontally.

We call the case that an element would be placed above a comparable greater element or below a lesser element, *overshooting*. However, to ensure that every intermediate drawing that is computed in the node step still satisfies the vertical invariant, we have to prohibit overshooting. Therefore, we add overshooting protection to the step in the algorithm where $(\vec{p}_a)_{a\in X}$ is recomputed. This is done by restricting the movement of every element such that it is placed maximally $\frac{c_{\text{vert}}}{10}$ below the lowest positioned greater element, or symmetrically above the greatest lower element. If the damping factor is chosen sufficiently small, overshooting is rarely required. This is, because our forces are defined such that the closer two elements are positioned the stronger they repel each other, see Figure 6.4.

All three forces are then consolidated into a single routine that is repeated at most K times or until the total stress falls below a constant ε, see Algorithm 6.13. The general idea of our forces is similar to the forces described in Freese's approach, as comparable elements attract each other and incomparable elements repel each other. However, we are able to get rid of the ranking function that fixes y-coordinate and thus have an additional degree of freedom by and to include the horizontal distance in the determination of the vertical positions. Furthermore, our forces are formulated in a general way such that the drawings can be computed in arbitrary dimensions. This overcomes getting stuck in local minima.

6.3.2 Line Step

While the goal of the node step is to get a good representation of the internal structure by optimizing on the proximity of nodes, the goal of the line step is to make the resulting drawing more aesthetically pleasing by optimizing distances between lines. Thus, in this step the drawing is optimized on three soft criteria. First, we want to maximize the number of parallel lines. Secondly, we want to achieve large angles between two lines that are connected to the same node. Finally, we want to have a high distance between elements and non-adjacent lines. We achieve a better fit to these criteria by applying a force-directed algorithm with three different forces, each optimizing on one criterion. While the previous step does not directly incorporate the path of the lines, this step incorporates those into its forces. Therefore, we call this step the *line step*.

The first force of the line step operates on lines (a,b) and (c,d) with $a \neq c$ and $b \neq d$ if their cosine distance is below a threshold c_{par}. The horizontal force

$$f_{\text{par}}((\vec{p}_a,\vec{p}_b),(\vec{p}_c,\vec{p}_d)) = -\left(1 - \frac{d_{\cos}((\vec{p}_a,\vec{p}_b),(\vec{p}_c,\vec{p}_d))}{c_{\text{par}}}\right) \cdot \left(\frac{(\vec{p}_b-\vec{p}_a)_x}{y_b-y_a} - \frac{(\vec{p}_d-\vec{p}_c)_x}{y_d-y_c}\right)$$

Algorithm 6.14 The Line Step of ReDraw

Input: Ordered set: $(X, \leq)$
Drawing $p = (\vec{p}_a)_{a \in X} \subseteq \mathbb{R}^d$
Constants $K \in \mathbb{N}$, $\varepsilon > 0$, $\delta > 0$, $c_{\text{par}} > 0$, $c_{\text{ang}} > 0$, $c_{\text{dist}} > 0$
Output: Drawing: $p = (\vec{p}_a)_{a \in X} \subseteq \mathbb{R}^d$

```
def line_step($O, p, d, K, \varepsilon, \delta, c_{\text{par}}, c_{\text{ang}}, c_{\text{dist}}$):
  $t = 1$
  while $t < K$ and $\max_{a \in X} \|F_a(t)\| > \varepsilon$:
    $A = \{\{(a,b),(c,d)\} \mid a \prec b, c \prec d, d_{\cos}((\vec{p}_a,\vec{p}_b),(\vec{p}_c,\vec{p}_d)) < c_{\text{par}}\}$
    $B = \{\{(a,c),(b,c)\} \mid (a \prec c, b \prec c) \text{ or } (c \prec a, c \prec b), d_{\cos}((\vec{p}_a,\vec{p}_c),(\vec{p}_b,\vec{p}_c)) < c_{\text{ang}}\}$
    $C = \{(a,(b,c)) \mid a \in X, b \prec c, d(\vec{p}_a,(\vec{p}_b,\vec{p}_c)) < c_{\text{dist}}\}$
    for $a \in X$:
      $F_a(t) = \sum_{\{(a,b),(c,d)\} \in A} f_{\text{par}}((\vec{p}_a,\vec{p}_b),(\vec{p}_c,\vec{p}_d)) + \sum_{(a,(b,c)) \in C} f_{\text{dist}}(\vec{p}_a,(\vec{p}_b,\vec{p}_c))$
               $- \sum_{\{(b,a),(c,d)\} \in A} f_{\text{par}}((\vec{p}_a,\vec{p}_b),(\vec{p}_c,\vec{p}_d)) - \frac{1}{2} \sum_{(b,(a,c)) \in C} f_{\text{dist}}(\vec{p}_a,(\vec{p}_b,\vec{p}_c))$
               $+ \sum_{\{(a,c),(b,c)\} \in B} f_{\text{ang}}((\vec{p}_a,\vec{p}_c),(\vec{p}_b,\vec{p}_c))$
    for $a \in X$:
      $\vec{p}_a$ = overshooting_protection($\vec{p}_a + \delta \cdot F_a(t)$)
    $t = t + 1$
```

operates on a and the force $-f_{\text{par}}((\vec{p}_a,\vec{p}_b),(\vec{p}_c,\vec{p}_d))$ operates to b. The result of this force is thus that almost parallel lines are moved to become more parallel. Note that this force becomes stronger the more parallel the two lines are.

The second force operates on lines that are connected to the same node and have a small angle, i.e., lines with cosine distance below a threshold c_{ang}.

Let (a,c) and (b,c) be such a pair then the horizontal force operating on a is given by

$$f_{\text{ang}}((\vec{p}_a,\vec{p}_c),(\vec{p}_b,\vec{p}_c)) = \left(1 - \frac{d_{\cos}((\vec{p}_a,\vec{p}_c),(\vec{p}_b,\vec{p}_c))}{c_{\text{ang}}}\right) \cdot \left(\frac{(\vec{p}_c - \vec{p}_a)_x}{y_c - y_a} - \frac{(\vec{p}_c - \vec{p}_b)_x}{y_c - y_b}\right).$$

In this case, once again the force is stronger for smaller angles; however, the force is operating in the opposite direction compared to f_{par} and thus makes the two lines less parallel. Symmetrically, for each pair (c,a) and (c,b) the same force operates on a. There are artifacts from f_{par} that operate against f_{ang} in opposite direction. This effect should be compensated for by using a much higher threshold constant c_{ang} than c_{par}, otherwise the benefits of this force are diminishing.

Finally, there is a force that operates on all pairs of nodes a and lines (b,c), for which the distance between the element and the line is closer then c_{dist}. The force

$$f_{\text{dist}}(\vec{p}_a, (\vec{p}_b, \vec{p}_c)) = \frac{1}{d(\vec{p}_a, (\vec{p}_b, \vec{p}_c))} \cdot \left((\vec{p}_a - \vec{p}_c) - \frac{(\vec{p}_a - \vec{p}_c) \cdot (\vec{p}_b - \vec{p}_c)}{(\vec{p}_b - \vec{p}_c) \cdot (\vec{p}_b - \vec{p}_c)} (\vec{p}_b - \vec{p}_c) \right)$$

is applied to a and $-f_{\text{dist}}(\vec{p}_a, (\vec{p}_b, \vec{p}_c))/2$ is applied to b and c. This results in a force whose strength is linearly stronger, the closer the distance $d(\vec{p}_a, (\vec{p}_b, \vec{p}_c))$. It operates perpendicular to the line and repels the node and the line.

Similar to the node step, all three forces are combined into a routine that is repeated until the remaining energy in the physical system drops below a certain stress level ε. Furthermore, a maximal number of repetitions K is fixed. We also once again include the overshooting protection as described in the previous section to make sure that the vertical invariant stays satisfied.

The line step that is described in this section is a computational demanding task, as in every repetition of the iterative loop the sets of almost parallel lines, small angles and elements that are close to lines have to be recomputed. To circumvent this problem on weaker hardware, there are a number of possible speedup techniques. First, the sets described above do not have to be recomputed every iteration, but can be cached over a few iterations. In Algorithm 6.14, these are the sets A, B and C. By recomputing those sets only every k-th iteration a speedup to almost factor k can be achieved. Another speedup technique that is possible is to only execute the line step in the last round. Both of these techniques however have a trade-off for the quality of the final drawing and are thus not further examined in this work.

6.3.3 Dimension Reduction

In the dimension reduction step, we compute a $(d-1)$-dimensional drawing from the d-dimensional drawing with the goal of reflecting the structural details of the original drawing like proximity and angles. Our approach to solve this is to compute a $(d-1)$-dimensional linear subspace of the d-dimensional space. By preserving the vertical dimension we can ensure that the vertical invariant stays satisfied. Then, a parallel projection into this subspace is performed.

As such a linear subspace always contains the origin, we center our drawing around the origin. Thereby, the whole drawing $(\vec{p}_a)_{a \in X}$ is geometrically translated such that the mean of every coordinate becomes 0. The linear subspace projection is performed as follows: The last coordinate of the linear subspace will be the vertical component of the d-dimensional drawing to ensure that the vertical invariant is preserved. For the other $(d-1)$ dimensions of the original space, a principle component analysis [96] is performed to reduce them to a $(d-2)$-dimensional subspace. By combining this projection with the vertical dimension, a $(d-1)$-dimensional drawing is achieved, that captures the structure of the original, higher-dimensional drawing and represents its structural properties.

It is easily possible to replace principal component analysis in this step by any other dimension reduction technique. It would be thinkable to just remove the first coordinate in each step and hope that the drawing in the resulting subspace has enough information encapsulated in the remaining coordinates. Also, other ways of choosing the subspace in which is projected could be considered. Furthermore, non-linear dimension reduction methods could be tried in order to achieve drawings. However, our empirical experiments suggest, that principal component analysis hits a sweet spot. The payoff of more sophisticated dimension reduction methods seems to be negligible as each drawing is further improved in lower dimensions. On the other hand, we observed local minima if we used simpler dimension reduction methods.

6.4 Evaluation and Discussion

In this section, we evaluate and discuss the quality of the ReDraw algorithm. Thereby, we discuss its computational complexity, introduce the datasets we used for testing, and recommend a parametrization. Furthermore, we empirically demonstrate the quality of some exemplary drawings and present a user evaluation that we conducted to confirm the quality. We provide the source code [37] so that other researchers can conduct their own experiments and extend it.

6.4.1 Run-Time Complexity

The run-time of the node step is limited by $\mathcal{O}(n^2)$ with n being the number of elements, as the distances between every element pair are computed. The run-time of the line step is limited by $\mathcal{O}(n^4)$, as the number of lines is bounded by $\mathcal{O}(n^2)$. Finally, the run-time of the reduction step is determined by principal component analysis which is known to be bounded by $\mathcal{O}(n^3)$. Therefore, the total run-time of the algorithm is polynomial in $\mathcal{O}(n^4)$. This is an advantage compared to our approach in Chapter 7 and Sugiyama's framework, which both solve exponential problems. Still, it is noteworthy that, Sugiyama can be applied with a combination

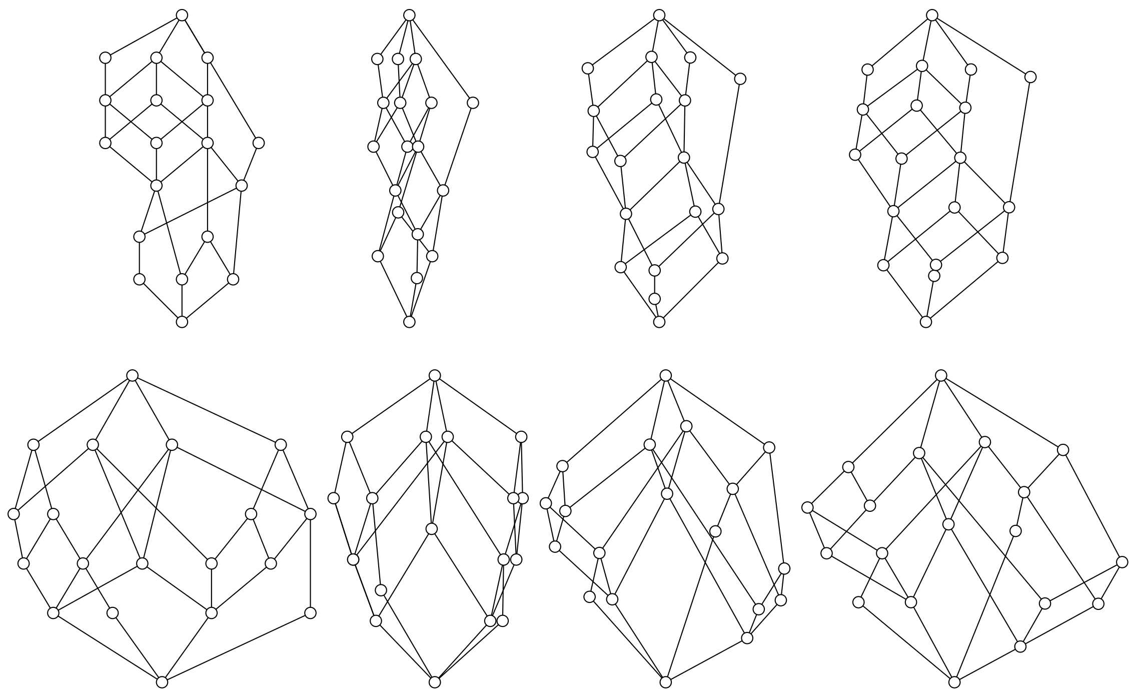


Figure 6.5: Drawing of the lattices for the formal contexts "forum romanum" (top) and "living beings and water" (bottom) from the test dataset. The algorithms to compute the drawings are Sugiyama (far left), Freese (left), ReDraw without the line step (right) and ReDraw (far right).

of heuristics and we also propose heuristics for our approach in Chapter 7 to overcome this problem. Freese's layout has by its nature of being a force-directed algorithm, similar to our approach, polynomial run-time.

6.4.2 Test Datasets

Our test dataset consists of 77 different lattices including all classical examples of lattices described in [57]. We enriched these by lattices of randomly generated contexts with attribute and object sets of cardinality between 5 and 10. Each incidence pair in these contexts is present with a probability of $\frac{1}{2}$. Furthermore, we sampled induced subcontexts from large binary datasets [92]. An overview of all related formal contexts for these lattices, together with their drawing generated by ReDraw is published together with its source code [37].

We restrict the test dataset to lattices, as lattice drawings are of great interest for the formal concept analysis community. This enables us to perform a user study using domain experts from the formal concept analysis community to evaluate the algorithm.

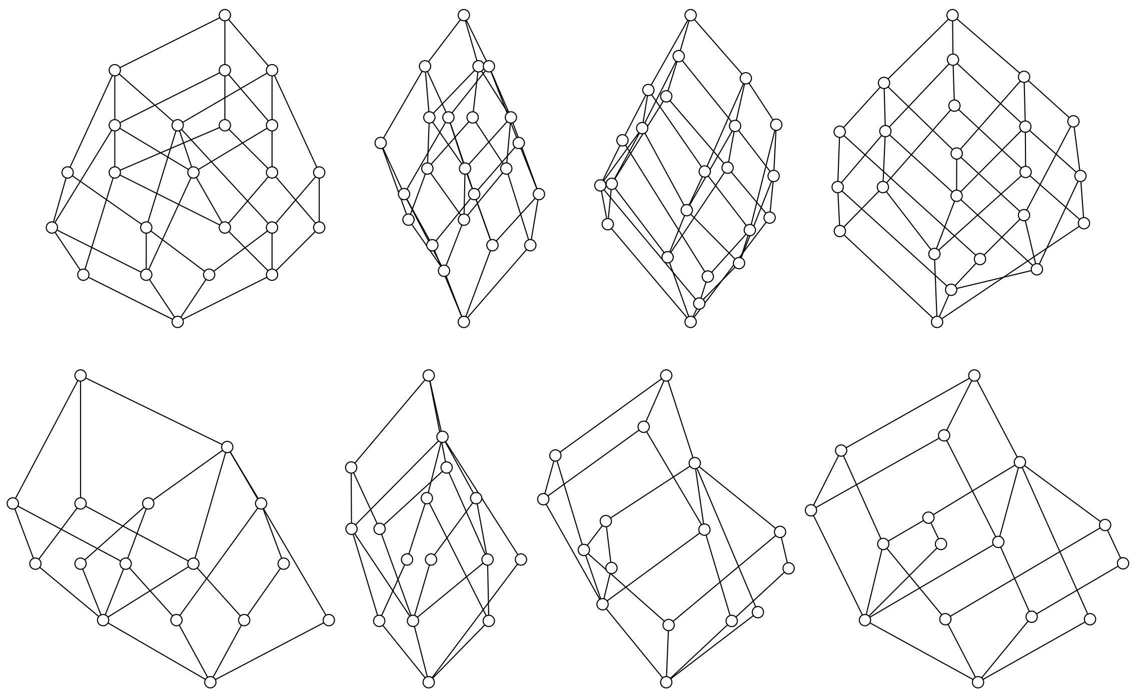


Figure 6.6: Drawing of the lattices for the formal contexts "therapy" (top) and "ice cream" (bottom) from the test dataset. The algorithms to compute the drawings are Sugiyama (far left), Freese (left), ReDraw without the line step (right) and ReDraw (far right).

6.4.3 Recommended Parametrizations

Our algorithm has multiple parameters that have to be chosen carefully, as they strongly influence the generated drawings. As it is hardly possible to conduct a user study for every single combination of parameters, our recommendations are based on empirical observations. We recommend the following parameters which we used throughout our evaluation. We used a maximal number of $K = 1000$ algorithm iterations or stopped if the stress in the physical system fell below $\varepsilon = 0.0025$. Our recommended damping factor $\delta = 0.001$. In the node step we set $c_{\text{vert}} = 1$ as the optimal vertical distance and $c_{\text{hor}} = 5$. We used the thresholds $c_{\text{par}} = 0.005$, $c_{\text{ang}} = 0.05$ and $c_{\text{dist}} = 1$ in the line step. The drawing algorithms are started with 5 dimensions as we did not observe any notable improvements with higher dimensional drawings. Finally the resulting drawing is scaled in horizontal direction by a factor of 0.5.

6.4.4 Empirical Evaluation

To demonstrate the quality of our approach, we compare the resulting drawings to the drawings generated by a selected number of different algorithms in Figure 6.5

and Figure 6.6. The different drawings are computed using Sugiyama's framework, Freese's layout, and our new approach. Additionally, a drawing of our approach before the line step is presented to show the impact of this line step. The approach proposed in this chapter achieves, in our opinion, satisfying results for these ordered sets. Modifications of ReDraw that combine the node step and the line step into a single step were tried; however, the then resulting algorithm did not produce the anticipated readability, as the node and line forces seem to conflict each other.

6.4.5 User Evaluation

As we described in the previous sections, it is not a trivial task to evaluate the quality of an order diagram drawing. Drawings that one human evaluator might consider as favorably might not be perceived as readable by others. To obtain a measurable evaluation, we conducted a user study to compare the different drawings generated by our algorithm to two other algorithms. We decided to compare our approach to Freese's and Sugiyama's algorithm, as those two seem to be the two most popular algorithms for lattice drawing at the moment. To reduce the chance of biases, we randomized the order in which the ordered sets are presented and shuffled the diagram algorithms presented in each step. Furthermore, we did not provide an allocation between the algorithms and the presented drawings.

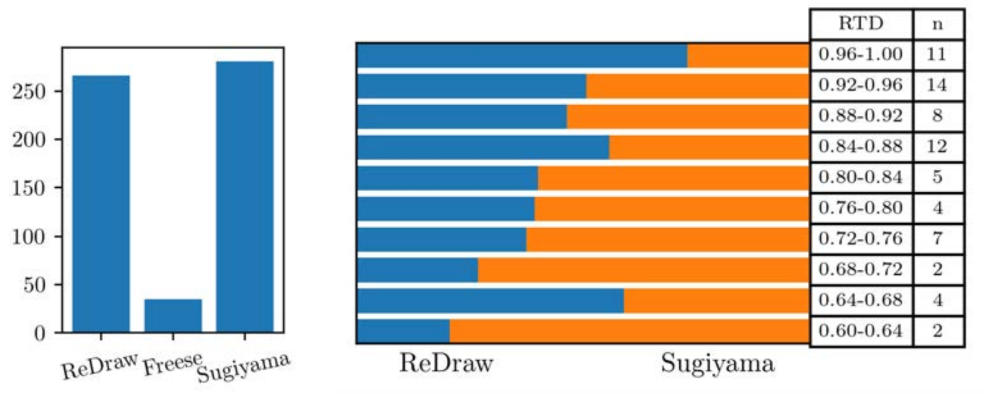
Experimental Setup

In each step of the study, all users are presented with three different drawings of one lattice from the dataset in random order and have to decide which one they perceive as “most readable”. The term “most readable” was neither further explained nor restricted.

Results

The study was conducted with nine experts from the formal concept analysis community to guarantee expertise with order diagrams among the participants. Thus, all ordered sets in this study were lattices. The experts voted 582 times in total; among those votes, 35 were cast for Freese's algorithm, 266 for our approach and 281 for Sugiyama.

A common property of lattices generated from real-world data is to have a high *truncated relative distributivity (RTD)* [114]. Thus, this property, which measures



RTD	n
0.96-1.00	11
0.92-0.96	14
0.88-0.92	8
0.84-0.88	12
0.80-0.84	5
0.76-0.80	4
0.72-0.76	7
0.68-0.72	2
0.64-0.68	4
0.60-0.64	2

Figure 6.7: Results of the user study. *Left:* Number of votes for each algorithm. *Right:* Share of votes for ordered sets divided into ranges of different truncated distributivity.

the share of distributive triples for a lattice excluding the bottom-element, is of special interest. Based on the RTD, we compared the share of votes for Sugiyama's framework and ReDraw for all order diagrams that are in a specific truncated distributivity range. The results of this comparison are depicted in Figure 6.7. The higher the RTD, the better ReDraw performs in comparison. The only exception in the range 0.64-0.68 can be traced back to a small test set with $n = 4$.

Discussion

As one can conclude from the user study, our force-directed algorithm performs on a similar level to Sugiyama's framework while outperforming Freese's force-directed layout. In our dataset, the ReDraw performs better, compared to Sugiyama's framework, on lattices that have a higher RTD. We observed similar results when we computed the relative normal distributivity, i.e., the same measure where the bottom element is not ommited. Thus, we recommend to use ReDraw for larger drawings that are highly distributive.

Furthermore, we can observe, that ReDraw performs better if there are repeating structures or symmetries in the lattice as each instance of such a repetition tends to be drawn similarly. This makes it the algorithm of choice for ordered sets that are derived from datasets containing high degrees of symmetries. Anyway, we are convinced that there is no single drawing algorithm that can produce readable drawings for all different kinds of order diagrams. It is thus always advisable to use a combination of different algorithms and then decide on the best drawing.

6.5 Conclusion

In this chapter we introduced our novel approach ReDraw for drawing order diagrams. Thereby, we adapted a force-directed algorithm to the realm of diagram drawing. In order to guarantee that the emerging drawing satisfies the hard conditions of order diagrams, we introduced a vertical invariant that was satisfied in every step of the algorithm. The algorithm consists of two main ingredients, the first being the node step that optimizes the drawing in order to represent structural properties using the proximity of nodes. The second is the line step that improves the readability for a human reader by optimizing the distances of lines. To avoid local minima, our drawings are first computed in a high dimension and then iteratively reduced into two dimensions. To make the algorithm easily accessible, we published the source code and gave recommendations for parameters. Generated drawings were, in our opinion, suitable to be used for ordinal data analysis in the context of formal concept analysis. A study using domain experts from this research area to evaluate the quality of the drawings confirmed this observation.

Further work in this realm could involve to modify the line step and combine it with different algorithms such as the structural approach that is introduced in the next section. Also, modifications to the procedure to produce additive drawings would be of great interest and should thus be investigated further.

CHAPTER 7

Order Diagrams Through Two-Dimension Extension

As we have seen before, easily readable drawings of order diagrams are hard to come by, even for small ordered sets. Although methods that transfer classical graph drawing techniques to order diagrams, such as the one presented in the previous chapter produce pleasing results for some ordered sets, they unfortunately do not perform to a satisfactory degree in general. One observation is that we generally do not use structural properties already encapsulated into the order for this process. This raises the following question.

(2.2) **How can we make use of structural properties of ordered sets to draw order diagrams?**

In this chapter, we present the algorithm DIMDRAW which, in contrast to classical graph drawing approaches, uses the encapsulated dimensional structure within the order to generate a drawing. Thereby, the ordered set is decomposed into linear orders which then give rise to an order diagram. The algorithm is based on a link between the dimension of an ordered set and the bipartiteness of a corresponding graph.

7.1 Introduction

The automatic generation of order diagrams is usually done by optimizing on a set of soft conditions such as minimizing the number of crossing lines or the number of different slopes. The algorithms produce drawings that are sufficiently readable, to some extent. Still, they may not compete with those created manually by an experienced human. However, such experts are often not available, too expensive, or not efficient enough to create large numbers of order diagrams.

In this section we thus follow another approach to propose an algorithm that we call DimDraw. We claim that it produces drawings that come close to the quality of hand-drawn line diagrams. In contrast to prior approaches it tries not to optimize on criteria but employs the dimensional order structure of the concept lattice itself. Recall, that any ordered set can be described as the intersection of linear orders and that the minimum number of linear orders, such that the intersection of these orders is again the ordered set is called the order dimension. We base our idea on the observation that ordered sets of order dimension two can be embedded into the plane naturally. Building up on this, we show a procedure to embed the ordered sets of order dimension three and above by reducing them to the two-dimensional case. To this end, we prove an essential fact about inclusion-maximal bipartite induced subgraphs in this realm. The emerging NP-hard computation problem can then be solved by the methods developed in Chapter 3. Our main contribution in this chapter with respect to this problem is Theorem 7.11.

We investigate our theoretical result on different real-world data sets using the just introduced algorithm DimDraw. The ordered set used for this experiment arise from of formal concept analysis and are evaluated through a user study of domain experts, as those are experienced in working with order diagrams. In this user study, the visualizations generated by the algorithm compete with two classical approaches of automated order diagram drawing.

7.2 Related Work

Our approach makes use of a natural embedding that arises from two-dimensional order diagrams. Two-dimensional orders are thereby embedded by their dominance drawing.

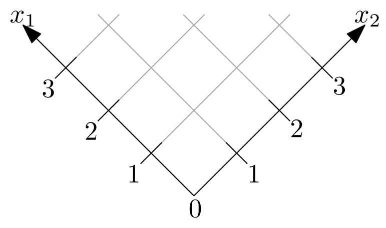


Figure 7.1: The grid that DIMDRAW embeds the order into.

Definition 7.1
A dominance drawing of an ordered set $(X, \leq)$ is a drawing where for each pair of elements $u, v \in X$ into two axes x_1, x_2 it holds that $x_1(u) < x_1(v)$ and $x_2(u) < x_2(v)$ if and only if $u < v$.

For guaranteeing that the upward property of order diagrams is preserved, the axes have to be embedded into the plane tilted by 45 degree, compare to Figure 7.1. By definition, a dominance drawing exists exactly if the ordered set is two-dimensional. Another characterization of all ordered sets with dominance drawings follows from a result of Baker et al.

Theorem 7.2 (Baker et al. [7])
An ordered set $(X, \leq)$ admits a dominance drawing, if and only if the set $(X \cup \{0, 1\}, \leq \cup (X \times 1) \cup (0 \times X))$ is planar.

Recall, that for some ground set the order $\leq_1$ is *conjugate* to another order $\leq_2$ if the comparability graph of one is the cocomparability graph of the other. The existence of a conjugate order is another criterion for the two-dimensionality.

Theorem 7.3 (Dushnik and Miller, 1941 [42])
The dimension of an ordered set $(X, \leq)$ is at most 2, if and only if there is a conjugate order $\leq_C$ on X. A realizer of $(X, \leq)$ is given by $\mathcal{R} = \{\leq \cup \leq_C, \leq \cup \geq_C\}$.

In 1977, Golumbic proposed an algorithm [60] to check whether a graph is transitive orientable, i.e., whether there is an order on its vertices, such that the graph is exactly the comparability graph of this order. It computes such an order, if it exists. This algorithm runs in $\mathcal{O}(n^3)$, with n being the number of vertices of the graph. Combining this with Theorem 7.3 provides an algorithm to compute whether an ordered set is two-dimensional and furthermore also returns a realizer, in this case. For the sake of completeness, note, that there are faster algorithms (as fast as linear) [89] for computing transitive orientations. However, those only work if the graph is actually transitive orientable and return erroneous results otherwise.

7.3 Drawing Ordered Sets of Dimension Two

Ordered sets of order dimension two have a natural way to be visualized using their dominance drawing. Let $(X, \leq)$ be an ordered set of dimension two. First, we need the notion of an element-position in a linear extension.

Definition 7.4
The *position* of a element $x \in X$ in a linear order $L = (X, \leq)$ is the number of vertices that are smaller, i.e.

$$\mathrm{pos}_L(x) := |\{y \in X \mid y < x\}|.$$

Now, let $\mathcal{R} = |\{\leq_1, \leq_2\}|$ be a realizer consisting of two linear extensions of the order relation $\leq$. The drawing of the corresponding lattice can be retrieved as follows: First, the vertices are placed in a coordinate system such that each element is embedded at the coordinates $(\mathrm{pos}_{\leq_1}(x), \mathrm{pos}_{\leq_2}(x))$. This coordinate system is then embedded into the plane using the generating vector $(-1, 1)$ for x_1 and $(1, 1)$ for x_2. Finally, the comparable elements in covering relation are connected by straight lines. A visualization, how the coordinate system is embedded into the plane can be seen in Figure 7.1. Each point now divides the plane into four quadrants using the two lines that are parallel to x_1 and x_2. It holds that $a < b$, if and only if the point b is in the quadrant above the point a by construction. The elements of the cover relation are drawn as straight lines. This guarantees that all elements of the cover relation are drawn as monotonically increasing curves.

In order to compute such drawings, a check of the two-dimensionality of the ordered set is required. If so, computing a realizer in polynomial time is possible.

Algorithm 7.15 Computation of the Conjugate Order

Input: Ordered set $(P,\leq)$
Output: Conjugate order of $(P,\leq)$

```
def Compute_Conjugate_Order(P,≤):
    C = Cocomparability_Graph(P,≤)
    if Has_Transitive_Orientation(C):
        (P,≤_C) = Transitive_Orientation(C)
        return ≤_C
    else:
        return ⊥
```

Algorithm 7.16 Dominance Drawing for Ordered Sets of Order Dimension Two

Input: Two-Dimensional Ordered Set $(X, \leq)$
Output: Dominance Drawing (v_x, v_y)

```
def dominance_drawing_2d($X$,$\leq$):
    $\leq_2$ = conjugate_order($X$,$\leq$)
    $l_1$ = $\leq \cup \leq_2$
    $l_2$ = $\leq \cup \geq_2$
    for $x$ in X:
        $v_x$ = $\text{pos}_{(X,l_1)}(x)$
        $v_y$ = $\text{pos}_{(X,l_2)}(x)$
    return ($v_x$,$y_x$)
```

This is done in Algorithm 7.15. The conjugate order can then be used to compute an embedding for a two-dimensional order as shown in Algorithm 7.16.

7.4 Projecting Drawings of Higher Dimension

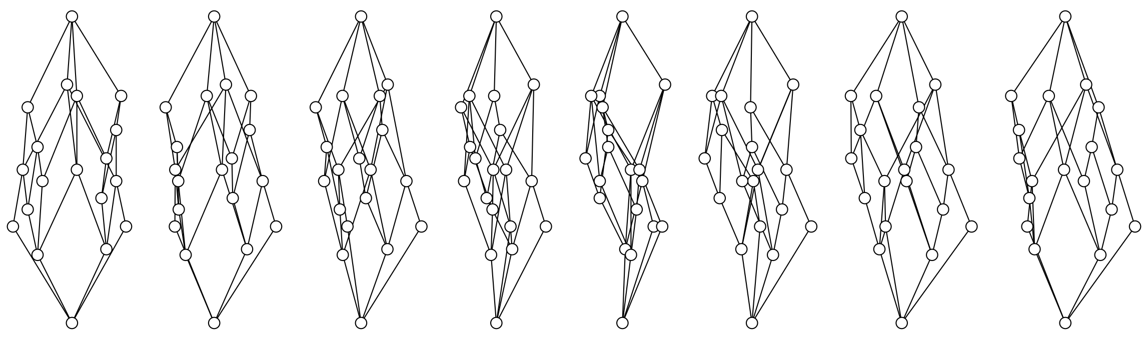


Figure 7.2: A three-dimensional ordered set embedded – based on its realizer – into three-dimensional Euclidean space and then projected into the plane using a parallel projection from multiple angles. Even though the structure of the ordered set is recognizable, the drawings are all not satisfactory. For better drawings refer to Figure 7.6.

The idea of embedding two-dimensional orders does generalize naturally to higher dimensions, i.e., gives us a nice way to embed n-dimensional ordered sets into the n-dimensional Euclidean space by using an n-dimensional realizer. However, this approach turns out to entail two problems. Deciding the order dimension of an ordered set is NP-complete for ordered sets of dimensions 3 or more [119]. This results in the fact that computing realizers of such ordered sets is computationally hard. Furthermore, finding projections of an ordered set from a higher dimension into the plane turns out to be difficult. See, for example, the projections of an ordered set in example Figure 7.2, where we performed a parallel projection using eight different angles between 0 and 180 degrees. None of the diagrams seems to be readable. For this reason our algorithm makes use of a different method to compute higher dimensional drawings, that is still based on the concept of using the realizer. Note that the n-dimensional embeddings as proposed in this section might be used as a starting point for the algorithm proposed in the previous chapter, where a high-dimensional embedding is required.

7.5 DimDraw for Higher-Dimension Orders

The main idea is the following: for a given order relation we want to insert some number of additional pairs to make it two-dimensional. This allows the resulting order to be drawn using the algorithm for the two-dimensional case. Afterwards, we remove all the inserted pairs. By construction, the property that if $a < b$, the point a is inserted below b is still preserved. However, for each inserted pair (a, b), we obtain two points in the drawing that are drawn as if a and b were comparable. Such drawings are sometimes called *weak dominance drawings* [77].

Definition 7.5 (Two-Dimension Extension)
Let $(X, \leq)$ be an ordered set. A set $\mathcal{C} \subseteq \text{inc}(X)$ is called a *two-dimension-extension* of $(X, \leq)$, if and only if $\leq \cup\, \mathcal{C}$ is an order on X and the ordered set $(X, \leq \cup\, \mathcal{C})$ is two-dimensional.

Such an extension always exists: a linear extension of dimension one always exists, while an order with exactly one incomparable pair has dimension two. Recall that we already considered this problem in Chapter 5 with, where we referred to a result from Felsner and Reuter in Theorem 5.21 that stated that deciding whether a two-dimension extension with k pairs exists is NP-complete. Similarly to the incompatibility graph in the Chapter 5, here is also a graph whose bipartiteness corresponds to the requested property.

Definition 7.6 (Transitive Incompatibility Graph)
For $(X, \leq)$ we denote the set of incomparable elements by $\text{inc}(X, \leq)$. Two elements $(a, b), (c, d) \in \text{inc}(X, \leq)$ are called *incompatible*, if their addition to $\leq$ creates a cycle in the emerging relation, i.e., if there is some sequence of elements $c_1, \ldots, c_n \in X$, such that each pair $(c_i, c_{i+1}) \in \leq \cup \{(a, b)\} \cup \{(c, d)\}$ with $i \in \{1, \ldots, n\}$ and $c_{n+1} = c_1$. We call the graph $((\text{inc}(X, \leq), E)$ with $\{(a, b), (c, d)\} \in E$, if and only if (a, b) and (c, d) are incompatible the *transitive incompatibility graph*, denote this graph by $\text{tig}(X, \leq)$.

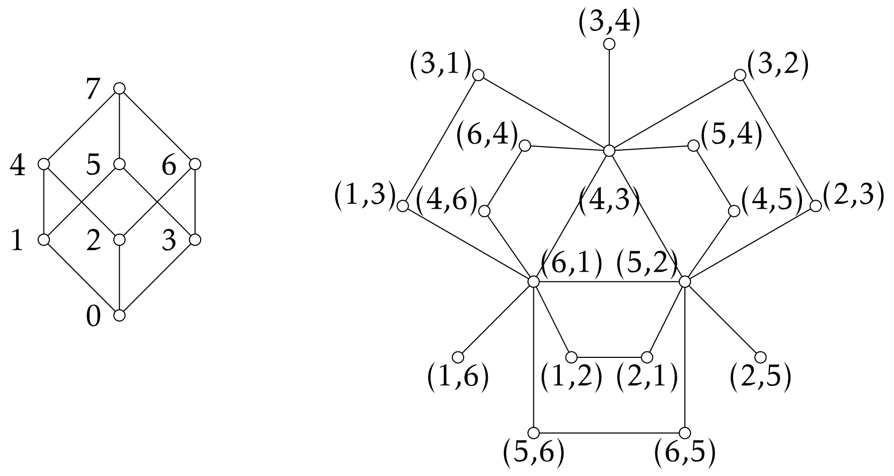


Figure 7.3: *Left:* The order diagram of the concept lattice of the contranominal scale of dimension three. *Right:* The corresponding transitive incompatibility graph.

The relationship between the bipartiteness of the transitive incompatibility graph and the two-dimensionality of its ordered set is given in the following.

Theorem 7.7 (Doignon et.al., 1984 [26])
The ordered set $(X,\leq)$ has order-dimension at most two if and only if $\operatorname{tig}(X,\leq)$ *is bipartite.*

An example of the transitive incompatibility graph is given in Figure 7.3. Note, that we cannot add arbitrary pairs to the order relation if we want it to stay an order. For example, if we add the pair $(2,1)$, we also have to add $(2,5)$ because of the transitivity. We formalize this dependency in the following.

Definition 7.8 (Enforcing Pair)
We say a pair $(a,b) \in \operatorname{inc}(X,\leq)$ *enforces* another pair $(c,d) \in \operatorname{inc}(X,\leq)$, if and only if $(c,d) \in (\leq \cup \{(a,b)\})^{+}$. If (a,b) enforces (c,d) in $\leq$, we use the notation $(a,b) \to (c,d)$.

Being an enforcing pair can also be characterized by the following theorem.

Lemma 7.9
If $(X,\leq)$ is an ordered set and $(a,b),(c,d) \in \operatorname{inc}(X,\leq)$, the following are equivalent:

i) $d \leq a$ *and* $b \leq c$.
ii) $(a,b) \to (d,c)$.
iii) $(c,d) \to (b,a)$.
iv) (a,b) *and* (c,d) *are incompatible.*

v) (b,a) and (d,c) are incompatible.

Proof. $(i) \Rightarrow (iv)$. Consider the relation $\prec := \leq \cup (a,b) \cup (c,d)$. This yields $d \prec a \prec b \prec c \prec d$, i.e, $\prec$ contains a cycle. Analogously, $(i) \Rightarrow (v)$.
$(iv) \Rightarrow (ii)$. The assumption directly implies that $(d,c) \in (\leq \cup (a,b))^+$ due to the cycle generated by (a,b) and (c,d). By the same argument $(v) \Rightarrow (iii)$.
$(ii) \Rightarrow (i)$. Assume $d \not\leq a$ or $b \not\leq c$. Since $(d,c) \in (\leq \cup (a,b))^+$ it follows $(d,c) \in \leq$ which contradicts $(c,d) \in \mathrm{inc}(X,\leq)$. Similarly follows $(iii) \Rightarrow (i)$. □

Recall the definition of $\mathrm{tig}(X,\leq)$ defined on $\mathrm{inc}(X,\leq)$ with incompatible pairs being connected. We call a cycle in $\mathrm{tig}(X,\leq)$ *strict*, if and only if for each two adjacent pairs (a,b) and (c,d) it holds that $d < a$ and $b < c$. A strict path is defined analogously.

Lemma 7.10 (Doignon et al., [26])
Let $(X,\leq)$ be an ordered set and let the pair $v \in \mathrm{inc}(X,\leq)$. Then the following statements are equivalent:

i) v is contained in an odd cycle in $\mathrm{tig}(X,\leq)$.
ii) v is contained in an odd strict cycle in $\mathrm{tig}(X,\leq)$.

This is stated implicitly in their proof of Proposition 2 [26], verifying the equality between the two chromatic numbers of a hypergraph corresponding to our cycles and a hypergraph corresponding to our strict cycles.

Recall, that it is NP-complete to decide whether an ordered set has a two-dimension extension. We hence propose an algorithm that tackles the problem for approximating the corresponding optimization problem using an idea that is based on the following. For $(X,\leq)$ of dimension greater than two, $\mathrm{tig}(X,\leq)$ is non-bipartite. Consider for example Figure 7.3. Because of the cycle of length 3 given by the pairs $(4,3)$, $(6,1)$ and $(5,2)$ in its transitive incompatibility graph, it can not have dimension two. Our approach is therefore to find a maximal induced bipartite subgraph in $\mathrm{tig}(X,\leq)$. This is in fact an order relation as the following shows.

Theorem 7.11
Let $(X,\leq)$ be an ordered set. Let $C \subseteq \mathrm{inc}(X,\leq)$ be minimal with respect to set inclusion, such that $\mathrm{tig}(X,\leq)\setminus C^{-1}$ is bipartite. Then $(X,\leq \cup C)$ is an ordered set.

Proof. Refer to the bipartition elements of $\mathrm{tig}(X,\leq)\setminus C^{-1}$ with P_1 and P_2.

Claim 1. The arrow relation is transitive, i.e., if $(a,b) \rightarrow (c,d)$ and $(c,d) \rightarrow (e,f)$ then $(a,b) \rightarrow (e,f)$. If $(a,b) \rightarrow (c,d)$ then $c \leq a$ and $b \leq d$ by definition. Similarly,

$(c,d) \to (e,f)$ implies $e \leq c$ and $d \leq f$. By transitivity of $\leq$ this yields that $e \leq a$ and $b \leq f$ which in turn implies that $(a,b) \to (d,f)$.

Claim 2. Let $(a,b),(c,d) \in \operatorname{inc}(X,\leq)$ with $(a,b) \notin C^{-1}$ and $(a,b) \to (c,d)$, then $(c,d) \notin C^{-1}$. Assume the opposite, i.e., $(c,d) \in C^{-1}$. Without loss of generality let $(a,b) \in P_1$. As $(c,d) \in C^{-1}$ there has to be a pair $(e,f) \in P_1$ that is incompatible to (c,d), i.e., $(e,f) \to (d,c)$, otherwise (c,d) can be added to P_1 without destroying the independet set. However $(a,b) \to (c,d)$ is equivalent to $(d,c) \to (b,a)$ and yields together with the transitivity of the arrow relation $(e,f) \to (b,a)$. But than (e,f) and (a,b) are incompatible, a contradiction since both are in P_1.

Claim 3. If $(x,y) \in C$, then $(y,x) \notin C$. As $(y,x) \in C^{-1}$, there is a pair (a,b) in P_1, such that (a,b) and (y,x) are incompatible, i.e., $(a,b) \to (x,y)$, otherwise (y,x) can be added to P_1. However, since $(a,b) \notin C^{-1}$ follows $(x,y) \notin C^{-1}$ directly from Claim 2.

Reflexivity: $\forall x \in X$ we have $(x,x) \in \leq \subseteq \leq \cup C$.

Antisymmetry: Assume $(x,y) \in \leq \cup C$ and $(y,x) \in \leq \cup C$. We have to consider three cases. First, $(x,y) \in \leq$ and $(y,x) \in \leq$. Then $x = y$, as $\leq$ is an order relation. Secondly, $(x,y) \in \leq$ and $(y,x) \in C$. If $(x,y) \in \leq$, then x and y are comparable, i.e., the pair (y,x) can't be in $\operatorname{inc}(P,\leq)$. Then $(y,x) \notin C$, a contradiction. Thirdly, $(x,y) \in C$ and $(y,x) \in C$. This may not occur because of Claim 3.

Transitivity: Let $(x,y) \in \leq \cup C$ and $(y,z) \in \leq \cup C$ we show $(y,z) \in \leq \cup C$. We have to consider four cases. First, $(x,y) \in \leq$ and $(y,z) \in \leq$ implies $(x,z) \in \leq$. Secondly, $(x,y) \in \leq$ and $(y,z) \in C$ and assume that $(x,z) \notin (\leq \cup C)$. Then $(z,x) \notin C^{-1}$, but $(z,x) \to (z,y)$, as $(x,y) \in \leq$. From Claim 2 follows that $(z,y) \notin C^{-1}$, a contradiction to $(y,z) \in C$. The case $(x,y) \in C$ and $(y,z) \in \leq$ is treated analogously. Lastly, $(x,y) \in C$ and $(y,z) \in C$ and assume $(x,z) \notin C$, i.e., $(z,x) \in C^{-1}$. There has to be an odd cycle in $P_1 \cup P_2$ together with (y,z), otherwise (y,z) can be added to $P_1 \cup P_2$ to create a larger bipartite graph. By Lemma 7.10, there also has to be a strict odd cycle. Let the neighbors of (y,z) in $P_1 \cup P_2$ be (a,b) and (c,d). Then $a < z$, $c < z$, $y < b$ and $y < d$, and the pairs (a,b) and (c,d) are connected by a strict path on an even number of vertices through the strict odd cycle. By the same argument there are pairs (e,f) and (g,h) with $e < y$, $g < y$, $x < f$ and $x < h$ and (e,f) and (g,h) connected by a strict odd path. We now show, that (z,x) is in an odd cycle with $P_1 \cup P_2$ to yield a contradiction. For this consider the following paths $A = (c,f)(z,x)(h,a)$, $B = (d,h)(h,g)$ and $C = (f,e)(e,b)$. Each of those is a path in $\operatorname{tig}(P,\leq)$ by definition.

Claim 4. Between (h,a) and (d,h) there is a path on an even number of vertices in $P_1 \cup P_2$. To show this, let $(a_1,b_1),\dots,(a_{2k},b_{2k})$ be the strict path on an even number of vertices connecting (a,b) and (d,c) such that $(a_1,b_1) = (a,b)$ and $(a_{2k},b_{2k}) = (d,c)$. This implies $a_{2i+1} < b_{2i+2}$, $a_{2i} < b_{2i+1}$, $b_{2i+1} > a_{2i+2}$ and $a_{2i} > b_{2i+1}$ and for each $i \in \{0,\dots,k-1\}$. However, this yields the path $(h,a) = (h,a_1)(b_2,h)(h,a_3)\cdots(b_{2k},h) = (d,h)$ which is even and connecting (h,a) and (d,h) in $P_1 \cup P_2$, as required.

Analogously, we obtain a path between (c,f) and (b,f) on an even number of vertices. Moreover (h,g) and (f,e) are also connected by a path on an even number of vertices in $P_1 \cup P_2$, since (g,h) and (e,f) are connected by an even path. Reversing all pairs of this path yields the required path. Combining the segments A, B and C with the paths connecting them yields an odd cycle in $P_1 \cup P_2 \cup \{(z,x)\}$, a contradiction. □

7.5.1 The Importance of Inclusion-Maximality

Consider the standard example $S_3 = (X,\leq)$ where the ground set is defined as $X = \{a_1,a_2,a_3,b_1,b_2,b_3\}$ and $a_i \not\leq a_j$ for $i \neq j$, $b_i \not\leq b_j$ for $i \neq j$ and $a_i \leq b_j$ if and only if $i \neq j$. This example is well known to be a three-dimensional ordered set. However, it becomes two-dimensional by inserting a single pair (a_i,b_i) into the order relation $\leq$ for some index $i \in \{1,2,3\}$, i.e., the transitive incompatibility graph becomes bipartite if we remove for example the pair (b_1,a_1). Now assume we do not require to remove a set minimal with respect to set inclusion, take for example both pairs (a_1,b_1) and (b_1,a_1). However the set $(X, \leq \cup \{(a_1,b_1),(b_1,a_1)\})$ is not an ordered set, as both pairs (a_1,b_1) and (b_1,a_1) are in $\leq \cup \{(a_1,b_1),(b_1,a_1)\}$. This is a conflict with $a_1 \neq b_1$, i.e., we do not preserve the order property as the resulting relation is not antisymmetric.

7.5.2 Finding Bipartite Subgraphs is Not Sufficient

From Theorem 7.11, one might conjecture that finding an inclusion-maximal bipartite subgraph of $\mathrm{tig}(X,\leq)$ is sufficient to find a two-dimension extension of $(X,\leq)$. However, it may occur that two pairs are not incompatible in $\mathrm{tig}(X,\leq)$ and are incompatible in $\mathrm{tig}(X,\leq \cup C)$ with C being an inclusion-minimal set such that $\mathrm{tig}(X,\leq)\backslash C^{-1}$ is bipartite. This can arise in particular, if the following pattern occurs: the ordered set contains the elements a,b,c and d, such that all elements are pairwise incomparable, except $b < d$ and $(c,a) \in C$ exhibits this observation, see Figure 7.4. Then (a,b) and (c,d) are not incompatible in $\mathrm{tig}(X,\leq)$, but they become incompatible with the relation $\leq \cup C$. An example for this is provided by

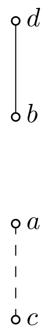


Figure 7.4: An example how new incompatibilities can arise. $\leq$ is the continuous line, C is the dashed line. (a,b) and (c,d) are not incompatible in $\leq$ and incompatible in $\leq \cup C$.

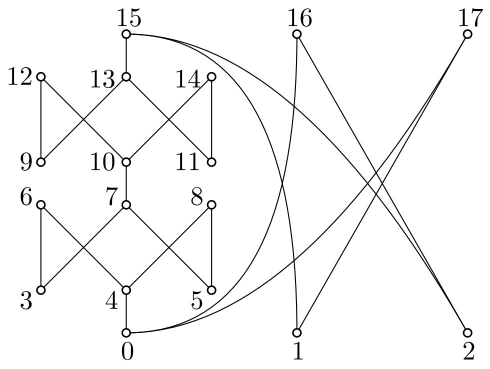


Figure 7.5: An example for an ordered set that has a transitive incompatibility graph with an inclusion-minimal induced bipartite subgraph of the transitive incompatibility graph which does not result in a two-dimension extension.

the ordered set in Figure 7.5. The transitive incompatibility graph of this ordered set has 206 vertices. By removing all the following pairs

```
(5,9), (3,13), (6,15), (17,15), (5,15), (5,13), (11,10), (9,6), (5,6),
(4,15), (1,8), (3,15), (11,12), (2,8), (11,7), (0,15), (1,16),
```

the transitive incompatibility graph of this ordered set becomes bipartite. However, if we once again compute the transitive incompatibility graph of the new ordered set, we see that it is not bipartite, i.e., the new ordered set is once again not two-dimensional. We have to add the additional pair `(17,8)` to make the graph two-dimensional. It may be remarked at this point that it is in fact possible to make the transitive incompatibility graph bipartite by adding only eleven pairs (in contrast to the seventeen added in this particular example), see Figure 7.10. Those pairs give rise to a two-dimension extension.

7.6 The DIMDRAW Algorithm

Building up on the ideas and notions from the previous sections, we propose the algorithm DIMDRAW as depicted in Algorithm 7.17. Given an ordered set $(X, \leq)$, one calls `Compute_Coordinates`. Until this procedure identifies a conjugate order using the `Comupte_Conjugate_Order` (and in turn the algorithm of Golumbic [60]), it computes the transitive incompatibility graphs using Lemma 7.9. It then identifies which vertices should be removed from the transitive incompatibility graph and adds the reverse pairs to the order $\leq$. To do so, it can use the algorithms

Algorithm 7.17 DIMDRAW Algorithm to Draw Ordered Sets

Input: Ordered set $(P,\leq)$
Output: Coordinates of the drawing of $(P,\leq)$

```
def Compute_Coordinates($P,\leq$):
    $\leq_C$ = Compute_Conjugate_Order($P,\leq$)
    $\mathcal{C} = \emptyset$
    while $\leq_C = \perp$:
        $I$ = Transitive_Incompatibility_Graph($P,\leq \cup C$)
        $B$ = Maximal_Bipartite_Subgraph($I$)
        $\mathcal{C} = \mathcal{C} \cup V(I \backslash B)^{-1}$
        $\leq_C$ = Compute_Conjugate_Order($P,\leq \cup C$)
    $\leq_1 = \leq \cup \leq_C$
    $\leq_2 = \leq \cup \geq_C$
    for $x$ in $P$:
        Coord($x$,1) = $|\{k \mid k \leq_1 x\}| - 1$
        Coord($x$,2) = $|\{k \mid k \leq_2 x\}| - 1$
    return Coord
```

proposed in Chapter 3. For finite ground sets, the algorithm terminates after finitely many steps.

7.6.1 Postprocessing

The algorithm does not prevent a point from being placed on top of lines connecting two different points. This however is not allowed in order diagrams. Possible strategies to deal with this problem are the following. One strategy is to modify the coordinate system, such that the marks of different integers are not equidistant. Another one is to perturb the points on lines slightly. A third way is to use splines for drawing the line in order to avoid such crossings. Finally, one might apply the line step introduced in the previous chapter to generate more parallel lines and more pleasing angles.

7.6.2 Runtime Discussion

Our algorithm has to solve an NP-complete problem. For this, it uses the also NP-complete problem of finding maximal induced bipartite subgraphs. To solve this problem, we proposed several algorithms in Chapter 3. If the SAT-version is used for this problem our algorithm will always take exponential time. Nonetheless, we give a short analysis of the algorithm, in case the algorithm is paired with one of the heuristics. Let for this purpose n be the number of elements of the ordered set. The

`Compute_Conjugate_Order` routine makes use of the `Transitive_Orientation`-Algorithm proposed by Golumbic [60] and runs therefore in $\mathcal{O}(n^3)$ time. The order of the transitive incompatibility graph can be quadratic in n, if every pair of elements is incompatible. Computing this graph can be done in $\mathcal{O}(n^2)$ time using *Lemma* 7.9. Therefore, in each repetition, we have to solve a problem with complexity at least $\mathcal{O}(n^2)$. By our experience from experiments the algorithm terminates after a small number of repetitions, i.e., at most two or three.

7.7 Evaluation and Discussion

To evaluate the quality of our algorithm, we tested it on all lattice examples from the standard literature book on formal concept analysis [57]. In all instances, the quality of the produced drawings seemed to reflect the underlying structure, similarly to the hand drawn versions of experts. For example, consider the lattice that arises from "Living Beings and Water" [57, p.18]. In Figure 7.6, we compare the hand-drawn example (left) to our algorithm (right). For Figure 7.7 [57, p.40], there are two different solutions depicted, both having the minimal number of pairs inserted, note that the algorithm stops after it finds a single solution. Because of the importance of drawings in formal concept analysis, we tested the algorithm on every lattice with eleven or less vertices. The reader might want to have a look at the document containing all 44994 drawings on 7499 pages [32].

As it is however hardly possible to come up with a metric that quantifies a generated order diagram to be close to a hand-drawn drawing, we also conducted a user study. In this study, in each step the user was presented with three drawings of the same ordered set, one being generated by DIMDRAW, one by Sugiyama's framework and one using Freese's algorithm. The user then had to decide which drawing was perceived as most readable. To ensure that no algorithm was at an advantage based on the positioning of the drawing, the order of the drawings was randomized in each step. Furthermore, the user was not told which drawing corresponds to which algorithm. Finally, the order in which the ordered set were presented to each user was also randomized. The data set we compiled for this user study consists of 100 formal contexts from real world data as well as randomly generated contexts. For reproducibility, we published the data set at [29], together with a description how every formal context in this set was collected.

Again, we restricted the user study to experts of the formal concept analysis community, as those are domain experts and experienced in working with order

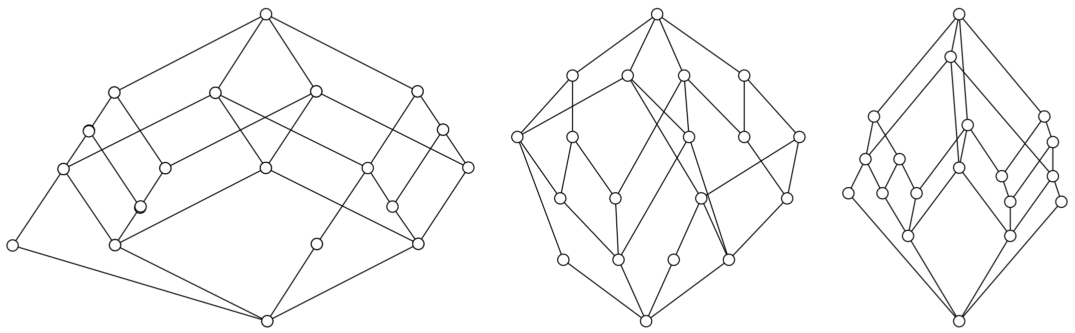


Figure 7.6: Three drawings of the "Living Beings and Water" lattice, by hand (left), with Sugiyama's framework (middle), and DIMDRAW (right).

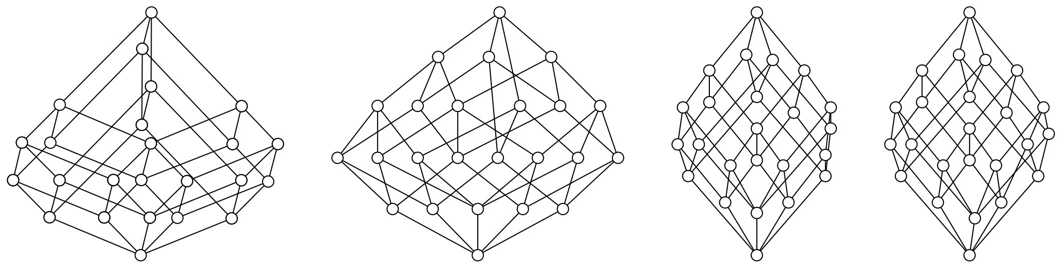


Figure 7.7: The "Drive concepts for motorcars" lattice. Drawn by an expert (far left), with Sugiyama's framework (middle left) and the two drawings of DIMDRAW (middle and far right).

diagrams. A total of 42 such domain experts participated in the study, each rating 37.95 drawings on average, yielding a total of 1594 ratings. The results of this study broken down to the individual drawings is published at [33]. Off all the 1594 ratings, 1030 were in favor of DIMDRAW, 510 in favor of Sugiyama's framework and 54 in favor of Freese, compare to Figure 7.8. We also compare for each ordered set of the study which drawing was perceived as best by the majority of users, i.e., which drawing got the most votes for each ordered set in Figure 7.9. Here, the drawing generated by DIMDRAW won 73 times, while Sugiyama's framework and the Freese layout won 24 times and 1 time respectively. The remaining two drawings were tied between DIMDRAW and Sugiyama. Thus, it can be concluded, that the quality of the drawings generated by DIMDRAW outperformed both, Sugiyama and Freese, in this user study.

Concluding the experiments, we want to present an example that our algorithm also works on non-lattices. Consider the ordered set from Figure 7.5. While the hand-drawn version of this order diagram makes use of splines, our algorithm-generated version Figure 7.10 uses exclusively straight lines.

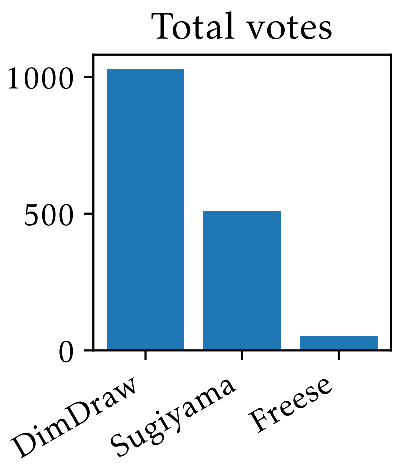


Figure 7.8: Total votes in the user study: A drawing generated by DimDraw was voted 1030 times as the best drawing, a drawing by Sugiyama's framework was voted 510 times and Freese's drawings were voted 54 times.

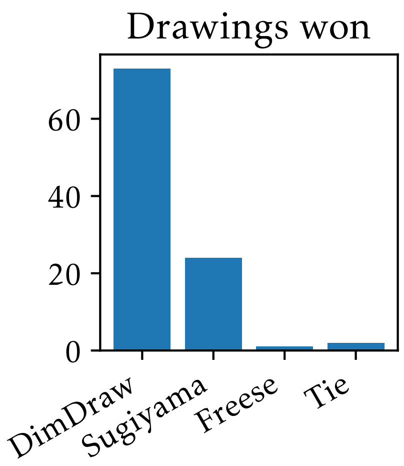


Figure 7.9: Drawings won by each algorithm: Of all 100 ordered sets, DimDraw's was 73 times voted to be the best drawing, Sugiyama's 24 times and Freese's once. The remaining two times Sugiyama and DimDraw tied.

An interesting observation during the experiments was the following: for all examples of that we are aware, even including those not presented in this work, one pass of the SAT solver was sufficient for reducing the order dimension to two. This is surprising, in particular in light of Figure 7.5 from Section 7.5.2 and poses an open problem, related to Open Problem 5.26.

Open Problem 7.12
Is there an ordered set $(X, \leq)$, such that an induced bipartite subgraph of maximal size of the transitive incompatibility graph does not give rise to a two-dimension extension?

Once again, this is of special interest, because it would allow DimDraw to compute a minimal two-dimension extension if combined with the SAT-solver, as the following shows.

Theorem 7.13
Let $(X, \leq)$ be an ordered set and $C \subseteq \operatorname{inc}(X, \leq)$ be of minimal cardinality, such that $\operatorname{tig}(X, \leq) \backslash C^{-1}$ is bipartite. If $(X, \leq \cup C^{-1})$ has order dimension two, $\leq \cup C^{-1}$ is a two-dimension extension of $\leq$ of minimal cardinality.

Proof. Assume there is a two-dimension extension $\tilde{C}$ of smaller cardinality. But then $\operatorname{tig}(X, \leq) \setminus \tilde{C}^{-1}$ is bipartite and of higher cardinality than $\operatorname{tig}(X, \leq) \setminus C^{-1}$, a contradiction. □

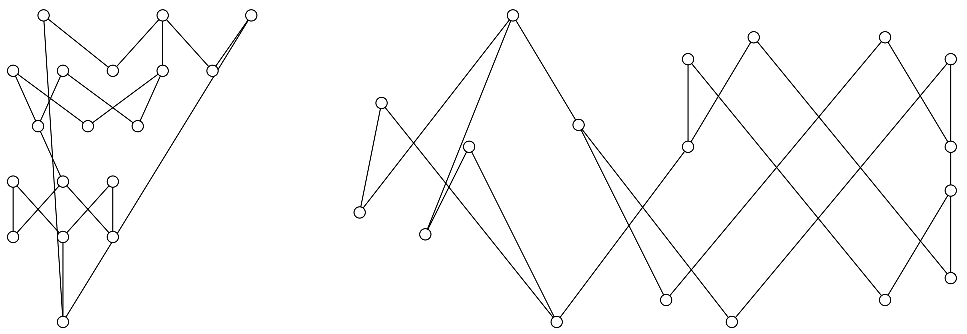


Figure 7.10: A non-lattice example of an ordered set. One drawing is produced by Sugiyama's framework (left) and on by DimDraw (right). See Figure 7.5 for a hand-drawn version.

7.7.1 Additional Drawings

In this section, we provide some additional drawings generated by the DimDraw algorithm for some standard datasets that are classical for formal concept analysis.

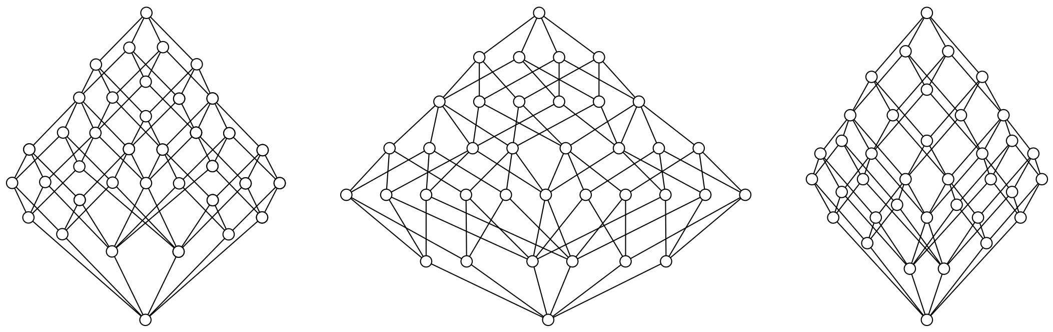


Figure 7.11: An example from [57, p.53]. The hand-drawn version is on the left, Sugiyama's framework in the middle, DimDraw on the right.

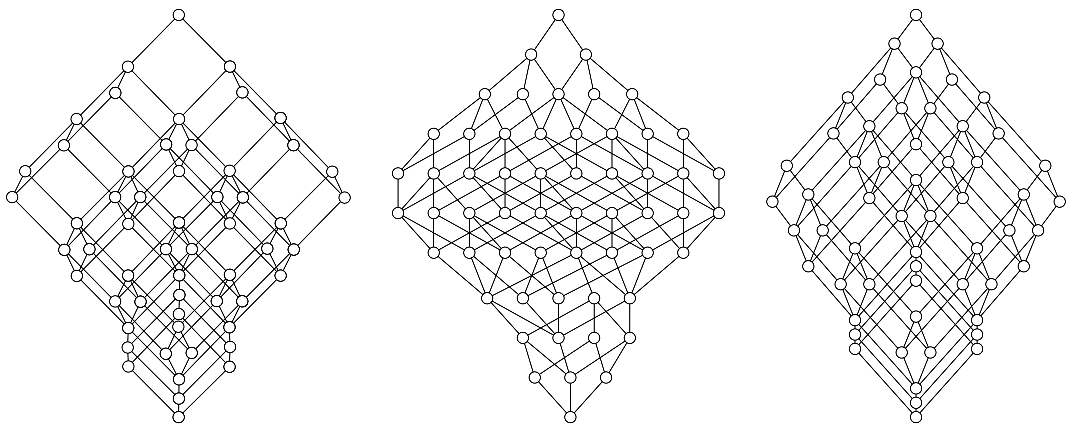


Figure 7.12: An example from [57, p.35]. The hand-drawn version is on the left, Sugiyama's framework in the middle, DimDraw on the right.

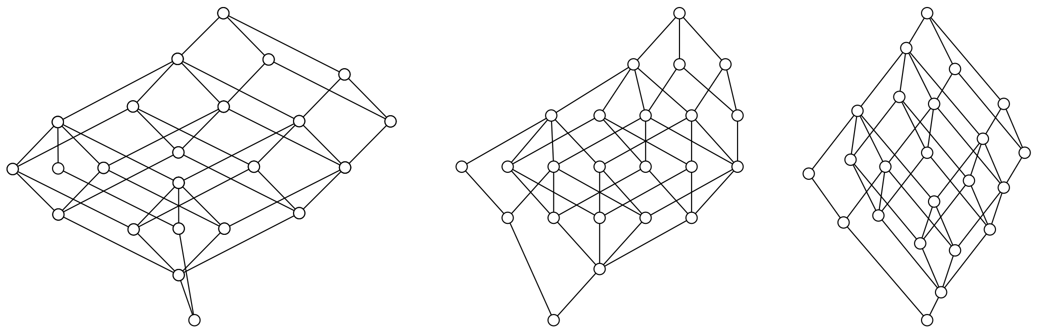


Figure 7.13: An example from [57, p.30]. The hand-drawn version is on the left, Sugiyama's framework in the middle, DIMDRAW on the right.

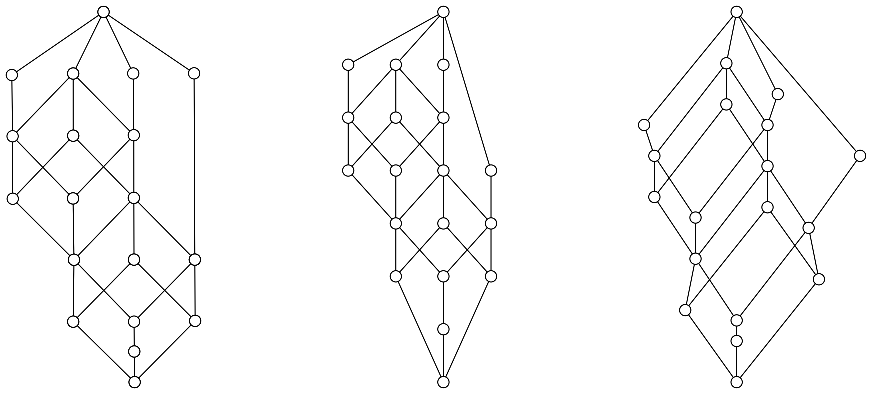


Figure 7.14: An example from [57, p.45]. The hand-drawn version is on the left, Sugiyama's framework in the middle, DIMDRAW on the right.

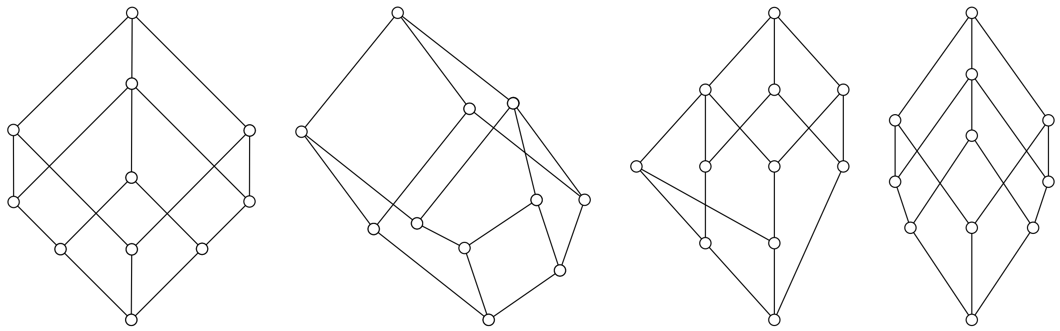


Figure 7.15: Order diagram of the lattice used in [53]. From left to right: Hand drawn, Ganter's algorithm, Sugiyama's framework, DIMDRAW.

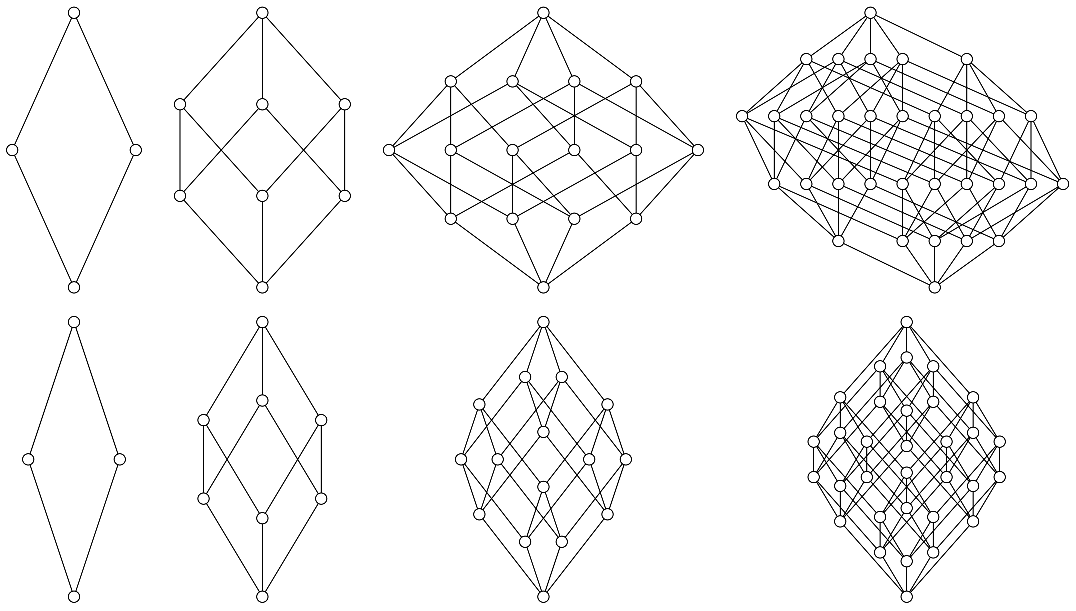


Figure 7.16: Order diagrams for Boolean lattices of dimension two, three, four and five drawn by Sugiyama's framework (top) and DimDraw (bottom).

7.8 Connection to Ordinal Factor Analysis

There is an interesting connection between the drawing approach as introduced in this chapter and the ordinal two-factorizations from Chapter 5. Recall that an ordinal factor is given by a *Ferrers relation* $F \subseteq G \times M$, where for all $g, h \in G$ and $m, n \in M$ the following condition holds: $(g, m) \in F$ and $(h, n) \in F \Rightarrow (g, n) \in F$ or $(h, m) \in F$. Following [57, Proposition 103], we know that $F \subseteq G \times M$ is a Ferrers relation if and only if $\mathfrak{B}(G, M, F)$ is a chain, i.e., all elements of $\mathfrak{B}(G, M, F)$ are pairwise comparable. A chain on the other hand corresponds to a linear order. Similarly to the order relation, there is the notion of the Ferrers relation which is strongly related to the order dimension.

Definition 7.14
The *Ferrers dimension* of a formal context (G, M, I) is the smallest number k, such that there exists a set of k Ferrers relations with their intersection being I.

The relation of the order dimension and the Ferrers dimension is given by:

Theorem 7.15 (Theorem 46[57])
Let $\mathbb{K} = (G, M, I)$ be a formal context. The Ferrers dimension of $\mathbb{K}$ is equal to the order dimension of its concept lattice.

As the complement to a Ferrers relation is once again a Ferrers relation, we furthermore know that the order dimension of $\mathfrak{B}(G,M,I)$ for some context (G,M,I) is at most d, if there are Ferrers relations $F_1,\ldots,F_d \subseteq G\times M$ with $G\times M\setminus I = \bigcup_{i=1}^{d} F_i$. This fact gives a handy connection and a way to decide the order dimension of a concept lattice. One has to cover all empty cells of the cross table of a concept with Ferrers relations. Note, that those do not have to be disjoint. Unfortunately, there is still the computational obstacle as deciding both, the order dimension and the Ferrers dimension, are NP-complete if the dimensions are three or higher.

We furthermore already showed in Lemma 5.23 that computing minimal two-dimension extensions and maximal ordinal two-factorization are related. Finally, the problems in both chapters related to the computation of two-dimensional substructures by finding maximal bipartite subgraphs, and we demonstrated in both cases that finding a maximal bipartite subgraph was not sufficient. In fact, the factor example in Section 5.3.5 is based on the ordered set $(X,\leq)$ from Section 7.5.2, as it is isomorphic to the formal context $(X,X,\not\leq)$. This also implies that the two Open Problems 5.26 and 7.12 are related.

7.9 Conclusion

We presented in this work a novel approach for drawing diagrams of order relations. To this end, we employed an idea by Doignon et al. relating order dimension and bipartiteness of graphs and proved an extension. We demonstrated various drawings in an experimental evaluation. The drawings produced by the algorithm were, in our opinion, satisfying. A user study with domain experts confirmed this observation. We would have liked to compare our algorithms to the heuristics developed in [78]. Unfortunately, we were not able to reproduce their results based on the provided description.

A notable observation is the fact that in all our experiments the SAT-Solver blend of DimDraw was able to produce a solution in the first pass, i.e., the algorithm found a truly minimal two-dimension extension. This raises the natural question, whether the maximal induced bipartite subgraph approach does always result in a minimal two-dimension extension. An implementation of DimDraw is included in the software conexp-clj [62].

Part IV

Vector Space Embeddings of Formal Concepts

CHAPTER 8

Learning Closure Operators

The task of representing large and high-dimensional data into low dimensional vector spaces is a necessary and essential task to computationally cope with contemporary datasets. Recent approaches like “word2vec” or “node2vec” are well established tools in this realm. We want to apply these approaches to the research realm of formal concept analysis and thus follow the lead of the following research question in this chapter.

(3) How can we leverage vector space embeddings of bipartite graphs to support knowledge discovery in formal concept analysis?

Thereby, we contribute to this line of research by introducing our approach CLOSURE2VEC, which is an embedding technique for formal concepts. This approach uses a neural network architecture to learn a closure operator.

8.1 Introduction

A common approach for the study of complex datasets is to embed them into a low dimensional real-valued Euclidean vector spaces $\mathbb{R}^d$, where d is small compared to the dimension of the original data. This enables the application of well understood and extensive methods from linear algebra. The underlying idea stems from the presumption that important features and connections scattered over many dimension in the data can be represented in few dimensions. Experience shows, that especially relative distances of the embedded entities are meaningful [90]. Furthermore, they are shown to be a successful tool in many research fields such as *link prediction* [61], *clustering* [5], and *information retrieval* [52]. The closest applications of such methods that come close to our work are performed by Wang et al. [109] and Ristoski et al. [93, 97] where knowledge graphs are embedded into low-dimensional vector spaces. We add to this line of research by embedding formal concepts using an idea motivated by word2vec. We demonstrate that the embedding is sensible and may be used to for the computation of structural connections in a formal context. Our embedding strategy deviates from the classical word2vec approach due to theoretical considerations. We revisit a previous work by Rudolph [99] and use this as a direction on how to propose a model for learning closure operators using neural networks. Contrary to Rudolph, who embeds the closure operator in a precise CLOSRUE2VEC is a learning approach.

To demonstrate the merit of our proposed method, we demonstrate how it can be used in order to rediscover (classical) FCA relevant features from data. In particular, we perform experiments on the covering relations and do an empirical evaluation. Finally, we present some first ideas on how to identify and extract implications using the learned closure operators. We thus discover that different aspects of closure operators can be encoded into a real-valued vector space embedding through a neural network technique. Because of the limited size of the datasets we deal with in formal concept analysis, we restrict ourselves to two and three dimensions. This has the additional benefit to align with the overall goal from FCA to create explainable methods, as such embeddings comprise the potential for human interpretability. or even explainability.

8.2 Related Work

In this chapter, we utilize embedding methods based on the shallow neural networks approach from word2vec [90] and its derivatives such as node2vec [61].

There have been previous attempts to link FCA and neural networks, such as using concept lattices to design network architectures for classification tasks in [82] and using a neural network to compute concept lattices in [16]. However, to the best of our knowledge, there has not been any research on embedding FCA closure systems into real-valued vector spaces using a learning approach based on neural networks. There have been numerous investigations into embedding finite ordinal data into real vector spaces, starting with measurement structures [100] which examined the basic feasibility of such an effort. Along this line of research in the field of FCA is a work of Wille [115], which focuses on ordinal formal contexts.

For FCA-based vector space embeddings, we are only aware of a work by Codocedo et al. [18] which computes real valued vector representations of objects and attributes using latent semantic analysis (LSA).

The research in the area of Resource Description Framework Graphs has been expanded upon by several works [97, 93, 109], which employ both basic and advanced methods. The main objective of these studies is to determine node similarity based on the relational structure. Our approach can be categorized as a similar approach. In order to uncover and utilize hidden conceptual relationships, we implement a neural network method for data in the form of formal contexts. Our approach builds on a study of Rudolph [99], where the author explores the use of neural networks in closure systems. Specifically, the author presents a way to precisely encode closure operators through neural networks using formal concept analysis.

8.2.1 Word2Vec

Throughout this chapter, the term embedding always refers to a general procedure or function, that maps some kind of structure (graphs, nodes, words) to a real-valued vector space. This notion is commonly used in machine learning and should not be confused with the strict embedding term from mathematics, which denotes a structure-preserving injection.

Our work is motivated by the idea of the word2vec approach [90, 91] where vector embeddings for words from large text corpora are generated. The underlying structure of the word2vec model is a neural network which is given as follows. A vocabulary $V = \{v_1, \dots, v_n\}$ is given as the set of words that are relevant for the training text. We map each word from the vocabulary to a unit vector of the vector space $\mathbb{R}^n$ by $\omega : V \to \mathbb{R}^n, v_i \mapsto e^i$, i.e., to the i-th vector of the standard basis of $\mathbb{R}^n$. We refer to this identification by the commonly used term *one-hot encoding*.

The neural network in word2vec than has the following structure. It consists of one input layer, then one hidden layer and an output layer with a softmax activation function, cf. Figure 8.1. The layer connections are dense. By construction, the first two layers thereby compose two linear functions. The first linear function maps the input from $\mathbb{R}^n$ to $\mathbb{R}^d$, the second one from $\mathbb{R}^d$ back to $\mathbb{R}^n$. Thus, the neural network can be expressed by a function from $\mathbb{R}^n \to \mathbb{R}^n$ that maps

$$x \mapsto \operatorname{softmax}(\psi(\varphi(x))),$$

where $\varphi : \mathbb{R}^n \to \mathbb{R}^d$ and $\psi : \mathbb{R}^d \to \mathbb{R}^n$ are both linear. They are composed by their corresponding matrices $W \in \mathbb{R}^{d\times n}$ and $U \in \mathbb{R}^{n\times d}$. The softmax activation of neural networks is given by

$$\operatorname{softmax} : \mathbb{R}^n \to \mathbb{R}^n, \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \mapsto \frac{1}{\sum_{l=1}^{n} \exp(x_l)} \begin{pmatrix} \exp(x_1) \\ \vdots \\ \exp(x_n) \end{pmatrix}.$$

The word2vec embeddings are then generated using the function φ.

For training the neural network of word2vec, the authors of the original paper propose [90] two different approaches to obtain the weights of the matrices W and U. In the *Skip-gram* model for a target word the context words around it are predicted, in the *Continuous Bag of Words* model, it is the other way round. The final embeddings are then generated as follows. Find for a given $d \in \mathbb{N}$ with $d \ll n$ the linear map $\varphi : \mathbb{R}^n \to \mathbb{R}^d$, i.e., a matrix $W \in \mathbb{R}^{d\times n}$ is used. It is assumed, that by this, words that appear in similar contexts are from this method mapped closely. The final embedding vectors of all words of the vocabulary are given by the map

$$\Upsilon : V \to \mathbb{R}^d, v \mapsto \varphi(\omega(v)).$$

8.2.2 Exact Representation of the Closure Operator

Thanks to a previous work by S. Rudolph [99], we are aware that it is possible to represent any closure operator on a finite set into a neural network function using

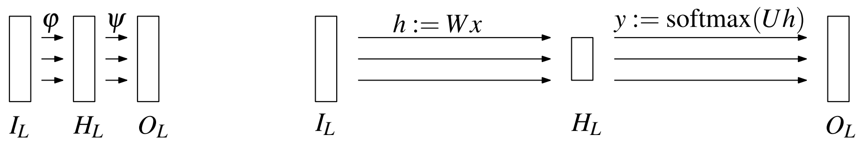


Figure 8.1: *Left:* A generic neural network consisting of 3 layers. *Right:* The strucutre of the word2vec architecture. The neural network consists of an input and output layer of size n and a hidden layer of size d, where $d \ll n$.

formal concept analysis. The network, as proposed in [99], consists of an *input layer* $I_L := \{0,1\}^{|M|}$, a *hidden layer* $H_L := \{0,1\}^{|G|}$ as well as an *output layer* called $O_L := \{0,1\}^{|M|}$. The mapping between I_L and H_L is defined as $\varphi = t \circ w$ consisting of a linear mapping w with transformation matrix $W = (w_{jh}) \in \{-1,0\}^{|M|\times|G|}$, such that

$$w_{jh} := \begin{cases} 0 & \text{if } (g_j, m_h) \in I, \\ -1 & \text{otherwise,} \end{cases}$$

and a non-linear activation function $t : \mathbb{R}^{|G|} \to \mathbb{R}^{|G|}$ with each component being mapped using the function $\tilde{t} : \mathbb{R} \to \{0,1\}$ defined as

$$\tilde{t}(x) = \begin{cases} 1 & x = 0, \\ 0 & x < 0. \end{cases}$$

The mapping between H_L and O_L is defined analogously by $\psi = \hat{t} \circ \hat{w}$, where once again $\hat{w}$ is a linear mapping with transformation matrix $\hat{W} = W^T$. The function $\hat{t} : \mathbb{R}^{|M|} \to \mathbb{R}^{|M|}$ is once again defined component-wise with each component being $\tilde{t}$. Using this construction, the function $\varphi \circ \psi$ encapsulates the closure operator. To find the closure of some attribute set $B \subseteq M$, one has to compute $\varphi \circ \psi$ of its binary encoding. Similar to both derivation operators of formal concept analysis, the mappings φ and ψ compute the attribute- and the object derivation, both in their binary encoding.

8.3 CLOSURE2VEC

Our goal is to employ the ideas from word2vec to improve the computational feasibility of common tasks in the realm of formal concept analysis. Doing so, we analyze different approaches and finally settle with a novel embedding technique that can provide more efficient computations. In particular, we consider the FCA problems of finding the covering relation of the concept lattice structure and the rediscovering of canonical bases. The linchpin of our investigation is the encapsulation of the closure operator of a formal context.

Analogue to the approach of word2vec, we want to achieve a meaningful embedding of the closure operator into $\mathbb{R}^d$ for $d = 2$ or $d = 3$. For the rest of this part, we assume that both the attribute set and the object set are indexed, i.e., for some context (G, M, I), we denote the object set by $G = \{g_1, g_2, \dots\}$ and the attribute set by $M = \{m_1, m_2, \dots\}$. This enables the possibility for defining the *binary encoding* of an object set A as the vector $v \in \{0,1\}^{|M|}$, with the $v_i = 1$ if and only if $m_i \in A$. Dually this can be done for attribute sets.

8.3.1 Considering the Unconstrained Problem

Considering the well established word2vec architecture, the following idea seems intuitive. Take the neural network layers as defined in Rudolph's architecture, but replace the hidden layer by a layer containing either two or three dimensions, i.e., we have $H_L = \mathbb{R}^d$. Instead of presetting φ and ψ, as in the last section, we want to retrieve them through machine learning. However, it may be noted that it is not meaningful to allow arbitrary and unconstrained functions. To see this, consider the following example with $d = 1$. Let $s : \{0,1\}^{|M|} \to \mathbb{N}$ be an injective mapping from the set of binary vectors of length $|M|$ to the natural numbers. Naturally, there is an inverse map $s^{-1} : s[\{0,1\}^{|M|}] \to \{0,1\}^{|M|}$, where $s[\{0,1\}^{|M|}]$ denotes the image of s of the domain. Since $\mathbb{N}$ is contained in H_L, we may find a natural continuation of s^{-1} to $\mathbb{R}$ by $\overline{s^{-1}} : \mathbb{R} \to \{0,1\}^{|M|}$ such that $\overline{s^{-1}}(x) := s^{-1}(\lfloor x \rfloor)$. Furthermore, let cl be the double application of the derivation operator, i.e., $(\cdot')'$ in the binary encoding. Using this setup let $\varphi := s$ and $\psi := \overline{s^{-1}} \circ cl$. Using these functions one can see that even though the neural network is able to compute the closure operator, the layer H_L contains no information about the formal context. This suggests that the set of possible functions has to be further constrained.

8.3.2 Representing Closures Using Linear Functions

Rudolph's approach for representing a closure operator by a neural network function entails two non-linear activation functions. This, however, is incompatible with the neural network approach proposed by word2vec, as this requires a linear map φ from I_L to H_L, which is also the final embedding we are looking for in our work. Note, that it is not possible to represent a closure operator using a linear function, since the closure of the empty set is not necessarily an empty set. The same fact is true for affine mappings, as showed in the following.

	1	2	3
a		×	×
b	×		×
c		×	

Figure 8.2: A formal context counterexample for Proposition 8.1. There is no affine mapping that maps each attribute set to its closure.

Proposition 8.1
Let (G,M,I) be a formal context. The set of all affine linear mappings, which represent the closure operator on the attribute set in binary encoding, can be empty.

Proof. Consider the formal context from Figure 8.2. For the sake of simplicity we speak about attribute and object sets and their respective binary encodings interchangeably. Assume that there is an affine mapping which maps each attribute set to its closure. Then there is a linear mapping l such that for each attribute set $v \in \{0,1\}^{|M|}$, the vector $v' = [1\ v]$ is mapped to the closure of v. Here $[1\ v]$ denotes the vector which results from the concatenation of a single bit (valued 1) with v. Using this one can infer that from

$$\begin{aligned}
\{\}'' &= \{a,b,c\}' = \{\}\\
\{3\}'' &= \{a,b\}' = \{3\}\\
\{1,2\}'' &= \{\}' = \{1,2,3\}\\
\{1,2,3\}'' &= \{\}' = \{1,2,3\}
\end{aligned}$$

follows that

$$\begin{aligned}
l(1,0,0,0) &= (0,0,0)\\
l(1,0,0,1) &= (0,0,1)\\
l(1,1,1,0) &= (1,1,1)\\
l(1,1,1,1) &= (1,1,1).
\end{aligned}$$

However, as l is a linear mapping, it is required that the following holds.

$$\begin{aligned}
l(1,1,1,1) &= l(1,1,1,0) + l(1,0,0,1) - l(1,0,0,0)\\
&= (1,1,1) + (0,0,1) - (0,0,0)\\
&= (1,1,2).
\end{aligned}$$

Hence, we obtain a contradiction. □

Corollary 8.2
Let (G,M,I) be a formal context. The set of all affine linear mappings, which represent the derivation operator on the attribute set in binary encoding, can be empty.

Proof. Assume there is such an affine linear map a. By duality we know that there must be an affine linear map a^d on the object set. A suitable composition of those mapping, i.e., using augmentation, contradicts with Proposition 8.1. □

8.3.3 Linear Representable Part of Closure Operators

We know from the last section that it is neither possible to represent the closure operator nor the derivation operator using an affine linear function. Still, it might be possible to obtain a meaningful approximation of an embedding using a linear map. In order to obtain some empirical evidence if studying this approach is fruitful we conduct a short experiment. Consider the neural network architecture as depicted in Figure 8.1 (left). Furthermore, let the input layer I_L of size $|M|+1$ be connected to a hidden layer H_L of low dimension, i.e., two or three, by a linear function φ. The layer H_L is connected to the output layer O_L that is of dimension M using a function ψ that consists of a linear function together with a *sigmoid* activation function. The first bit of I_L is always set to 1 and therefore a so-called bias unit. For our experiment, we now train the neural network by showing it randomly sampled attribute sets as inputs and their attribute closures as output, both in their binary encoding. We employ for this mean squared error as the loss function and a learning rate of 0.001. Even though the neural network starts to memorize the samples, it has seen after around 20 epochs, it does not generalize to attribute sets not previously seen in training. Furthermore, the resulting embedding into $\mathbb{R}^d$ does empirically not expose a meaningful structure. Additionally, this observation does not alter by changing the function ψ to a linear function. Also, experiments in which we investigated learning only the derivative operator were not fruitful. This is the expected behavior from our considerations in the last section. We do not claim that there are no better performing approaches for this task. Still this result motivates our progression to a different task.

8.3.4 Non-linear Embedding through CLOSURE2VEC

As linear embeddings do not seem to work out for our learning task, we employ a different approach. For this, we define a distance function based on the closure operator as follows.

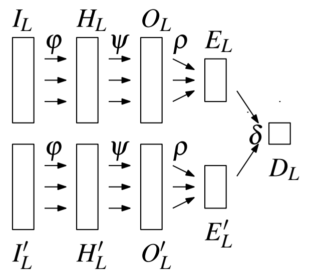


Figure 8.3: The siamese neural network used to train CLOSURE2VEC. The functions φ, ψ, ρ are shared functions between the layers in this model.

Definition 8.3 (Closure Hamming Distance (CHD))
Let the *closure Hamming distance (CHD)* for two attribute sets $A \subseteq M$ and $B \subseteq M$ be the distance function $d(A,B) := d_H((A'')^b, (B'')^b)$, where $\cdot^b$ denotes the binary representation and d_H is the Hamming distance.

Note, that the closure Hamming distance is not a metric, as the distance between two attribute sets sharing the same closure is 0, even though they are not the same. Based on the idea that two attribute sets are similar if they have a small CHD, we want to embed the attribute sets into a low-dimensional real-valued space, i.e., two or three dimensions. The goal here is that the embedding is approximately an isometric map.

We train a neural network architecture that we call CLOSURE2VEC to learn the just introduced CHD. For this, consider the network depicted in Figure 8.3. It consists of two input layers I_L and I'_L, each of size $|M|$. Then the function φ consisting of a linear function and a *ReLU-activation* function (see [84]) is used to feed the data into the hidden layers H_L and H'_L respectively, both of size $|G|$. After this, the function ψ, consisting of a linear function and a ReLU-activation function is applied to the two "streams" in the network. The result then is the input for two output layers O_L and O'_L, both of size $|M|$. This, however is not the final step of this network model. Finally, the layers E_L and E'_L, which consist of either two or three dimensions, are fed from O_L and O'_L, respectively, via ρ. This function is again built via composing a linear function and another ReLU-activation function. The output layer D_L has size one. Using a fixed function δ (in our case either the Euclidean distance or cosine distance), we compute a distance between O_L and O'_L. By sharing the functions φ, ψ, and ρ between the different layers, we ensure that a commutation of the input sets does not lead to a different prediction of the neural network.

The network then is trained by showing it two attribute sets in binary encoding as well as their closure Hamming distance at the output layer. The required loss function for this setup is the mean squared error. The learning rate of our network is set to 0.001. The training set for our approach is sampled as follows: For some $t \in \mathbb{N}$ take all attribute combinations that contain at most t elements and put them in some set $\mathcal{X} = \{X_1, X_2, \dots\}$, hence, $X_i \subseteq M$. For each $X_i \in \mathcal{X}$ generate a random attribute $m_i \in M$. Let the set $\mathcal{Y} = \{Y_1, Y_2, \dots\}$ with

$$Y_i = \begin{cases} X_i \setminus \{m_i\} & \text{if } m_i \in X_i, \\ X_i \cup \{m_i\} & \text{else,} \end{cases}$$

and finally $Z = \{z_i := d(X_i, Y_i)/|M| \mid X_i \in \mathcal{X}, Y_i \in \mathcal{Y}\}$ as the set of pairwise closure Hamming distances. The network is trained by showing it the binary encodings of X_i, Y_i, and z_i. Note, that the values of Z are normalized.

8.4 Evaluation and Discussion

This section contains experimental evaluations for our research.

8.4.1 Datasets

We conduct experiments on two different datasets. We depict the statistical properties of these datasets in Table 8.1. A detailed description of each follows.

wiki44k. The first dataset we use in this work is the wiki44k data set taken from [65] and then adapted by [64]. It consists of relational data extracted from Wikidata in December 2014. Even though it is constructed to be a dense part of the Wikidata knowledge graph, it is relatively sparse for a formal context.

Mushroom. The Mushroom dataset [27, 92] is a well investigated and broadly used dataset. It consists of 8124 mushrooms and has twenty-two nominal features that are scaled into 119 different binary attributes to form a formal context. The Mushroom dataset, compared to wiki44k, is denser, and even though it has a smaller number of objects, contains 10 times the concepts of wiki44k.

8.4.2 FCA Features Through CLOSURE2VEC

To evaluate the embeddings produced by CLOSURE2VEC, we introduce two FCA related problems: computing the covering relation of a concept lattice and com-

Table 8.1: The different datasets used to evaluate CLOSURE2VEC.

	Wiki44k	Mushroom
Number of Objects	45021	8124
Number of Attributes	101	119
Density	0.04	0.19
Number of Concepts	21923	238710
Mean attributes per concept	7.01	16.69
Mean objects per concept	109.47	91.89
Size of the Canonical Base	7040	2323

puting the canonical base, which is a minimal base of implications, for a given formal context. The intention here is to rediscover structural features from FCA in low dimensional embeddings. We choose for the dimension two and three in order to respect our overall goal for human interpretability and explainability. We test two different functions δ for the distance between the output layers O_L, O'_L, cf. Section 8.3.4. More specifically, we employ the Euclidean distance and the cosine distance. We conduct our experiments on two larger than average sized formal contexts. Precisely, we test the Wiki44k[64] and the well investigated Mushroom dataset[27, 92]. A comparison of the statistical properties of datasets we use is depicted in Table 8.1.

Distance of Covering Relation

For this experiment, we first compute the set of all concepts $\mathfrak{B}$ of a given formal context $\mathbb{K}$. The covering relation is an important tool for ordinal data science. Elements of the covering relation are essential for investigating and understanding order relations and order diagrams. However, in the case of large formal contexts computing the covering relation of the concept lattice can get computationally expensive, as this problem is linked to the transitive reduction of a graph [2].

The experimental setup now is as follows. First, we train the neural network architecture as introduced in Section 8.3.4. Hence, an input element is a binary encoded attribute X_i, another binary encoded attribute set Y_i of size $|X_i| \pm 1$, and the closure Hamming distance of them. In our experiment we fix $|X_i|$ to be four or less. Furthermore, we train the network over five epochs using the learning rate 0.001, with batch size 32, and mean-squared-error as loss function.

To evaluate the structural quality of the obtained embedding, we computed the covering relation of the concept lattices. In the following, we compare the dis-

Table 8.2: Distance between concept pairs as computed by CLOSURE2VEC, that are in covering relation (CR) and that are not in the covering relation (Non-CR).

Wiki44k:

Dim 2		Mean:	Std.:	**Dim 3**		Mean:	Std.:
Euc:	CR:	0.17	0.14	Euk:	CR:	0.16	0.15
	Non-CR:	0.71	0.59		Non-CR:	1.54	1.41
Cos:	CR:	0.63	0.33	Cos:	CR:	0.15	0.27
	Non-CR:	0.99	0.71		Non-CR:	0.36	0.43

Mushrooms:

Dim 2		Mean:	Std.:	**Dim 3**		Mean:	Std.:
Euc:	CR:	0.14	0.41	Euc:	CR:	0.13	0.29
	Non-CR:	0.51	0.38		Non-CR:	0.49	0.41
Cos:	CR:	0.49	0.43	Cos:	CR:	0.05	0.12
	Non-CR:	0.96	0.74		Non-CR:	0.18	0.22

tances between pairs of concepts in covering relation against concepts that are not in covering relation. The results of these experiments are depicted in Table 8.2. For all embeddings, the expected distance for two concepts in covering relation is significantly smaller than the expected difference of two concepts not in covering relation. This is true for both data sets. However, the observed effect is more notable for the wiki44k dataset. The Euclidean distance outperforms the cosine distance in all experiments using two and three dimensions.

Distance of Canonical Bases

In this experiment, we look at the canonical base of implications for a formal context and try to rediscover this canonical base in the computed embedding. The experiment consists of two different parts. The first part has the following setup. Take an implication of the canonical base, i.e., take (P,C) where $P \subseteq M$ is the premise and $C \subseteq M$ is the conclusion. For such an implication, construct all single conclusion implications (P,c) where $c \in C$. Then, compute the distance (with the same distance functions as used for the embedding process) of P and the embedding for all c. Additionally, also embed all $m \in M \setminus C$. Essentially, by doing so, we embed all $m \in M$ using our embedding function. We do this for all elements of the canonical base.

Now compute for all implications from the canonical base the following distances. First, the distances between a premise P and all its singleton conclusions c of (P,C). Secondly, the distances between P and $m \in M \setminus C$. Equipped with all

Table 8.3: Distances of implication premises and conclusions for singleton implications (S-Impl) and implications (Impl) in the embeddings computed by CLOSURE2VEC.

Wiki44k:

Dim 2		Mean:	Std.:	**Dim 3**		Mean:	Std.:
Euc:	S-Imp:	0.94	0.28	Euc:	S-Imp:	0.93	0.29
	Non-S-Imp:	0.73	0.40		Non-S-Imp:	0.83	0.47
	Imp:	0.44	0.33		Imp:	0.60	0.45
	Non-Imp:	0.51	0.48		Non-Imp:	0.65	0.55
Cos:	S-Imp:	1.00	0.69	Cos:	S-Imp:	0.69	0.53
	Non-S-Imp:	1.00	0.67		Non-S-Imp:	0.90	0.43
	Imp:	1.01	0.70		Imp:	0.93	0.56
	Non-Imp:	1.01	0.70		Non-Imp:	0.96	0.58

Mushrooms:

Dim 2		Mean:	Std.:	**Dim 3**		Mean:	Std.:
Euc:	S-Imp:	0.95	0.41	Euc:	S-Imp:	0.86	0.34
	Non-S-Imp:	0.45	0.32		Non-S-Imp:	0.42	0.27
	Imp:	0.70	1.01		Imp:	0.57	0.66
	Non-Imp:	1.02	0.98		Non-Imp:	0.88	0.68
Cos:	S-Imp:	1.00	0.65	Cos:	S-Imp:	1.05	0.36
	Non-S-Imp:	1.00	0.65		Non-S-Imp:	1.02	0.35
	Imp:	1.00	0.69		Imp:	0.88	0.53
	Non-Imp:	0.99	0.69		Non-Imp:	0.89	0.55

these distances, we try to detect a structural difference in favor of the embedded implications in contrast to other combinations of attribute sets, i.e., pairs of premises P and $m \in M \setminus C$. When using the cosine distance function we observe minimal structural difference. However, when using the Euclidean distance function, we detect a significant structural difference. In particular, for pairs of (P, c) with $c \in C$, the mean of the distances is significantly higher than the mean distance of the distances for some premise P with all singleton sets $m \in M \setminus C$. The observation is even stronger in the case of the Mushroom data set when compared to wiki44k. The results are depicted in Table 8.3 by the rows S-Imp (for the combinations (P, c)) and Non-S-Imp (for combinations (P, m)).

For the second part, we embed both attribute sets, i.e., the premise P and the conclusion C, for an implication from the canonical base. For every pair we compute the distance of P and C and compare them to the distance between two randomly generated attribute sets X, Y with $|X| = |P|$ and $|Y| = |C|$. The later is set for reasons of comparability. As shown in Table 8.3 we again detect structural differences for the considered implications and the randomly generated sets using average distances as features. In fact, the distance between the two randomly generated sets is on average larger than the average distance between premises and conclusions from implications drawn from the canonical base.

Discussion

In both experiments concerning the closure system embedding, we are able to rediscover and infer conceptual structures in the embeddings. Generally, we find that it is favorable to use for the Euclidean distance case the squares of the closure Hamming distances as output, i.e., for z_i. Overall, we discovered a significant bias in distances of embedded concepts that are in covering relation. This signal is even stronger for the wiki44k dataset. We suspect that this can be attributed to the lesser density of this dataset compared to Mushroom. The observations can be exploited naturally for mining covering relations, or important parts of those, from embedded concept lattices.

For the second experiment, we can report that the neural network embedding of parts of the closure system allows for rediscovering implicational structures. Since we trained our neural network on attribute sets of size four and smaller, we were interested in the number of closures our algorithm encounters. For both datasets, we can report that this number is approximately 10% of all closures.

The described behavior differs between the experiment where we train the network using whole implications, i.e., premise and conclusions, and single-conclusion implications. In cases where an attribute is element of the conclusion of some canonical base implication, the distance to the premise set is significant. At this point, we are unable to provide a rationale for that. The same goes for the second part of the experiment where we compare premise-conclusion pairs of canonical base implications with randomly generated pairs of attributes having the same sizes. We are not yet aware how this observation can improve the computations of the canonical base. This would need a more fundamental investigation of bases of implicational theories with respect to closure system embeddings. For example, one has to investigate if the observed effect is also true for other kinds of bases, e.g., direct basis [1]. As a final remark, we report that the Euclidean distance performed in all our experiments better than the cosine distance for both problem settings.

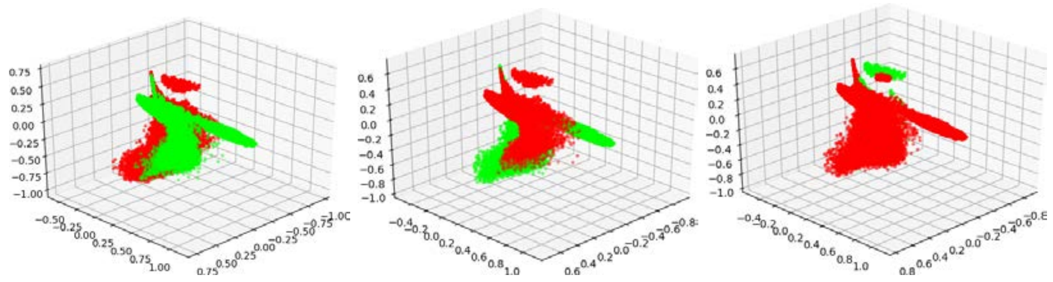


Figure 8.4: Embedding of all concepts of the mushroom dataset into three dimensions using CLOSURE2VEC. The coloring is done as follows. *Left:* The edible mushrooms are green, the non-edible mushrooms are red. *Middle:* The mushrooms with a broad gill are green, the mushrooms with a narrow gill are red. *Right:* The mushrooms with a crowded gill spacing are green and the mushrooms with a distant gill spacing are red.

Empirical Structural Observations

In addition to the two experiments above, we want to provide some insights we discovered for conceptual structures in our embeddings. We note that concepts sharing attributes seem to result in meaningful clusters. To see this, one can consider Figure 8.4. There, we see the same embedding of all formal concepts of the Mushroom dataset in three dimensions. In each of the three subfigures we colored different sets of concepts in red and green. In the first (Figure 8.4, left), we depict in red the not edible mushrooms and in green the edible ones. Even though we employed a very low dimensional embedding, we can still visually identify the two different classes. Hence, our embedding approach preserved some structure. The same seems true for the other depictions in which we colored broad gill versus narrow gill and crowded gill spacing versus distant gill spacing. Therefore, we are confident that our approach for low dimensional embeddings of closure systems using neural networks is beneficial. Moreover, as this empirical study shows, the low dimensional representation is still visitable by a human data analyst.

8.5 Conclusion

In this chapter, we presented CLOSURE2VEC, a first approach for modeling data and conceptual structures from FCA to the realm of embedding techniques based on word2vec. Taken together, the ideas in this chapter outline an appealing connection between formal concepts, closure systems, low dimensional embeddings, and neural networks learning procedures. We are confident that future research may draw from the demonstrated first steps. In our investigation, we have found convincing theoretical as well as experimental evidence that FCA-based methods can profit from word2vec-like embedding procedures. We demonstrated that closure operator embeddings that result from simple neural network learning algorithms may capture significant portions of the conceptual structure. Furthermore, we were able to demonstrate that the covering relation of the set of formal concepts may be partially extracted from a low dimensional embedding.

All these results were achieved while obeying the constraint for human interpretable and/or explainable embeddings. Applying neural network learning procedures on large and complex data does not necessarily constitute a contradiction to explainability when combined with conceptual notions from FCA.

Part V

Conclusion

CHAPTER 9

Summary

In this chapter, we briefly recap the techniques that we developed previously in this thesis. We relate the content of the different chapters to each other and illustrate how they tie in with the general direction that we followed in this work which is to enable a comprehensive and explainable analysis of bipartite graphs. Thereby, we developed tools that enable a human to perform structural investigations and knowledge extraction on such datasets. We designed the methods such that the emerging structures can be used to explain connections encapsulated in the data. As the research realm of formal concept analysis provides an elaborate toolkit of such methods, we decided to locate our work in this area. Formal concept analysis makes use of a canonical mathematical structure for the analysis of bipartite graphs which is the concept lattice. We leverage this lattice for our investigations.

The goal of Part II was the discovery of interesting substructures in data. In the first chapter of this part, we demonstrated how to discover large induced bipartite subgraphs within graphs. To this end, we developed an exact approach to compute the largest induced bipartite subgraph and three heuristics with different time-performance trade-offs. By demonstrating that the all computed algorithms performed to a satisfactory degree, we were able to answer research question (1.1). We used these algorithms later in Chapters 5 and 7, where we related a bipartite subgraph to the dimensional structure of the data.

The shape of the concept lattice of a formal context is highly influenced by two substructures that can appear in the formal context, namely contranominal scales and Ferrers relations. Contranominal scales in a formal context have a strong influence on the number of concepts as they give rise to Boolean suborders. To answer research question (1.2), we proposed in Chapter 4 an algorithm to compute the set of all attribute combinations which entail contranominal scales. Based on these attribute subsets, the algorithm can then compute all contranominal scales in the formal context. The identification of such attribute combinations is highly important for understanding a dataset, as they only give rise to trivial implications. The second substructure that influences the shape of the concept lattice are Ferrers relations. A maximal Ferrers relation in the formal context corresponds to a chain, i.e., a linear order in the concept lattice. Approaching research question (1.3), we proposed two approaches to compute such Ferrers relations in Chapter 5. First, we presented an algorithm that can compute two large ordinal factors in a formal context. This two-factorization can be used for a visualization technique and thus enable humans to understand and explain connections within the data. Secondly, we provided a greedy strategy that starts by computing the largest ordinal factor, and then subsequently iterates the next largest, until the whole dataset is covered by factors. We then demonstrated how to use the knowledge encapsulated in this ordinal factorization and how it can be used to navigate through the data.

In Part III, we developed two algorithms for the visualization of concept lattices. The concept lattice is the canonical way to visualize relationships within a dataset in formal concept analysis. It precisely represents the formal context (or bipartite graph), as each object is positioned below an attribute if and only if the attribute and object pair is the incidence relation (or connected by an edge in the bipartite graph formulation). The concept lattice can furthermore be used for the extraction of implications between the attributes. Therefore, it is a powerful and thorough way to explain connections in the dataset and extract knowledge from it. To this end, it is essential that the concept lattice is easily readable and makes algorithms, such as the two proposed in the visualization part of the thesis, a necessity. For the adoption of a classical graph drawing, as requested by research question (2.1) we proposed a force-directed diagram drawing algorithm. It performs a physical simulation which moves a system to a state of minimal stress. We chose the forces of the algorithm in a way that optimizes several soft criteria which are known to improve readability. To overcome local minima, the algorithm starts with a drawing in a high-dimensional Euclidean space which we then subsequently decreased. The second algorithm which approaches to answer research question (2.2) uses a structural approach based on the following idea. There are special concept lattices, which posses a natural way to be embedded into the plane using their dimensional structure. We demonstrated how to transform each concept lattice with minimal alterations into such a nicely emendable lattice and then use the so-emerging embedding to embed the unaltered lattice.

Finally, in Part IV, we approached research question (3) by presenting an approach to embed the set of all formal concepts for a given context into a low-dimensional vector space. The motivating idea for this approach was that we assumed the existence of an embedding which entails and depicts structural properties of the concepts. To compute such an embedding, we modified the idea of the word2vec algorithm based on mathematical considerations. The computed embedding was then deduced from a trained neural network. To verify that the embeddings in fact represent some structural properties of the concept lattice, we verified that attributes in implications were embedded with closer proximity on average. We did the same investigation on the distance of concepts in covering relation and where able to demonstrate that the same is true for them. Finally, we demonstrated that the embeddings are sensible using an empirical approach. Still, we believe that this research question cannot be considered settled yet, as future research has to prove that the computed embeddings can benefit algorithms in formal concept analysis.

CHAPTER 10

Outlook

We believe that there are significant questions related to our research that are yet-to-be-answered. In this chapter, we discuss a few of these questions and give hints how to tackle the underlying problems. We see potential for future research in all three parts of this thesis, which we thus present individually. Furthermore, we believe that it is possible to compose the different parts of the thesis into a comprehensive data analysis framework. To this end, we envision a data analysis tool that allows a human, who is not necessary well-versed in mathematics or knowledge discovery, to investigate bipartite graph datasets. In such a tool, it might be desirable to also include more methods that were developed in the context of formal concept analysis, such as attribute exploration.

In Part II, we proposed algorithms to discover substructures in relational data. The very first chapter of this part proposed algorithms to compute induced bipartite subgraphs of (non-bipartite) graphs. While better algorithms are always desirable, our proposed algorithms seem to work to a satisfying degree. In this realm, it would be interesting to see how modifications of our algorithms would perform for other hereditary graph properties.

The second and third chapter of the same part then deal with the discovery of two substructures that determine the structure of bipartite graphs, the contranominal scales and the Ferrers relations which are closely related to ordinal scales. Another interesting structure that we did not consider in this thesis, but stands out in this regard, is the nominal scale. The nominal scale is a formal context where each data point appears only if the other data points are absent. We believe that developing algorithms for the discovery of nominal scales is very interesting. Furthermore, we suspect that every bipartite graph can be built up from a hierarchy of nested contranominal scales, ordinal scales and nominal scales. Defining, investigating, and applying this hierarchy of building blocks is in our opinion a very interesting task for future work.

We proposed two algorithms for the visualization of concept lattices in Part III. An apparent next step in this regard could be the combination of the two approaches. It is straight-forward to apply the line-step from the ReDraw algorithm to drawings that are generated by the DimDraw algorithm. On the other hand, it seems promising to use the realizer that emerges from the order dimension for the initial embedding of the ReDraw algorithm. The initial embedding could in this case be computed using the technique that is employed in DimDraw. In the previous paragraph, we proposed the investigation of a structural hierarchy of building blocks for bipartite graphs. We believe that this hierarchy could be strongly related to the order-dimensional structure of the concept lattice. Thus, it might be possible to use this structure to compose a drawing algorithm, possible relying on the idea of the DimDraw algorithm to embed the different hierarchical levels. Finally, we are convinced that the whole area of order diagram and graph drawing would benefit from the existence of a well-tested evaluation function which measures the readability of a graph drawing. Such a function could replace the human evaluators that are required for the development of new tools, at least in the early stages.

In the final part of the thesis, we proposed a way to embed the set of all concepts into a vector space. While we demonstrated that the emerging embeddings comprise promising structure, we did not yet give a method that makes use of the information provided. In this realm, we are interested in the development of algorithms that leverage the so-proposed algorithms. We could, for example, imagine that it is possible for algorithms that compute implications to leverage the proposed embeddings. Furthermore, we could imagine that the computed embeddings can be used for visualization purposes, as our experimental evaluation demonstrated that the embeddings seem to be sensible by human standards.

Part VI

Appendix

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Adaricheva, K. V., Nation, J. B., and Rand, R. "Ordered direct implicational basis of a finite closure system." In: *Discret. Appl. Math.* 161.6 (2013), pp. 707–723. DOI: 10.1016/j.dam.2012.08.031.

[2] Aho, A. V., Garey, M. R., and Ullman, J. D. "The Transitive Reduction of a Directed Graph." In: *SIAM J. Comput.* 1.2 (1972), pp. 131–137. DOI: 10.1137/0201008.

[3] Albano, A. "Polynomial growth of concept lattices, canonical bases and generators: extremal set theory in formal concept analysis." PhD thesis. Universität Dresden, 2017.

[4] Albano, A. and Chornomaz, B. "Why Concept Lattices Are Large - Extremal Theory for the Number of Minimal Generators and Formal Concepts." In: *Proceedings of the Twelfth International Conference on Concept Lattices and Their Applications, Clermont-Ferrand, France, October 13-16, 2015*. Ed. by Yahia, S. B. and Konecny, J. Vol. 1466. CEUR Workshop Proceedings. CEUR-WS.org, 2015, pp. 73–86. URL: https://ceur-ws.org/Vol-1466/paper06.pdf.

[5] Alshari, E. M., Azman, A., Doraisamy, S., Mustapha, N., and Alkeshr, M. "Improvement of Sentiment Analysis Based on Clustering of Word2Vec Features." In: *28th International Workshop on Database and Expert Systems Applications, DEXA 2017 Workshops, Lyon, France, August 28-31, 2017*. IEEE Computer Society, 2017, pp. 123–126. DOI: 10.1109/DEXA.2017.41.

[6] Bachmaier, C., Gleißner, A., and Hofmeier, A. *DAGmar: Library for DAGs*. [Online; accessed 14-February-2023]. 2012. URL: https://www.infosun.fim.uni-passau.de/~chris/down/MIP-1202.pdf.

[7] Baker, K. A., Fishburn, P. C., and Roberts, F. S. "Partial orders of dimension 2." In: *Networks* 2.1 (1972), pp. 11–28. DOI: 10.1002/net.3230020103.

[8] Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999. ISBN: 0-13-301615-3.

[9] Bellman, R. "Dynamic programming." In: *Science* 153.3731 (1966), pp. 34–37.

[10] Belohlávek, R., Glodeanu, C. V., and Vychodil, V. "Optimal Factorization of Three-Way Binary Data Using Triadic Concepts." In: *Order* 30.2 (2013), pp. 437–454. DOI: 10.1007/s11083-012-9254-4.

[11] Belohlávek, R. and Vychodil, V. "Discovery of optimal factors in binary data via a novel method of matrix decomposition." In: *J. Comput. Syst. Sci.* 76.1 (2010), pp. 3–20. DOI: 10.1016/j.jcss.2009.05.002.

[12] Belohlávek, R. and Vychodil, V. "Formal Concepts as Optimal Factors in Boolean Factor Analysis: Implications and Experiments." In: *Proceedings of the Fifth International Conference on Concept Lattices and Their Applications, CLA 2007, Montpellier, France, October 24-26, 2007*. Ed. by Eklund, P. W., Diatta, J., and Liquiere, M. Vol. 331. CEUR Workshop Proceedings. CEUR-WS.org, 2007. URL: https://ceur-ws.org/Vol-331/Belohlavek1.pdf.

[13] Boeck, P. D. and Rosenberg, S. "Hierarchical classes: Model and data analysis." In: *Psychometrika* 53.3 (1988), pp. 361–381.

[14] Bron, C. and Kerbosch, J. "Finding All Cliques of an Undirected Graph (Algorithm 457)." In: *Commun. ACM* 16.9 (1973), pp. 575–576.

[15] Bundeszentrale für politische Bildung. *Wahl-O-Mat*. [Online; accessed 07-February-2022]. 2021. URL: https://www.bpb.de/politik/wahlen/wahl-o-mat/337541/download.

[16] Caro-Contreras, D. E. and Mendez-Vazquez, A. "Computing the Concept Lattice using Dendritical Neural Networks." In: *Proceedings of the Tenth International Conference on Concept Lattices and Their Applications, La Rochelle, France, October 15-18, 2013*. Ed. by Ojeda-Aciego, M. and Outrata, J. Vol. 1062. CEUR Workshop Proceedings. CEUR-WS.org, 2013, pp. 141–152. URL: https://ceur-ws.org/Vol-1062/paper12.pdf.

[17] Child, D. *The essentials of factor analysis*. Cassell Educational, 1990.

[18] Codocedo, V., Taramasco, C., and Astudillo, H. "Cheating to achieve Formal Concept Analysis over a Large Formal Context." In: *Proceedings of The Eighth International Conference on Concept Lattices and Their Applications, Nancy, France, October 17-20, 2011*. Ed. by Napoli, A. and Vychodil, V. Vol. 959. CEUR Workshop Proceedings. CEUR-WS.org, 2011, pp. 349–362. URL: `https://ceur-ws.org/Vol-959/paper24.pdf`.

[19] Cohen, S., Kimelfeld, B., and Sagiv, Y. "Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties." In: *J. Comput. Syst. Sci.* 74.7 (2008), pp. 1147–1159. DOI: 10.1016/j.jcss.2008.04.003.

[20] Community, T. G. D. *Rome and North Graphs*. [Online; accessed 14-February-2023]. URL: `https://www.graphdrawing.org/data.html`.

[21] Cook, S. A. "The Complexity of Theorem-Proving Procedures." In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*. Ed. by Harrison, M. A., Banerji, R. B., and Ullman, J. D. ACM, 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[22] Coombs, C. H. "A theory of data." In: (1964).

[23] Das, S., Sen, M. K., Roy, A. B., and West, D. B. "Interval digraphs: An analogue of interval graphs." In: *J. Graph Theory* 13.2 (1989), pp. 189–202. DOI: 10.1002/jgt.3190130206.

[24] Demel, A., Dürrschnabel, D., Mchedlidze, T., Radermacher, M., and Wulf, L. "A Greedy Heuristic for Crossing-Angle Maximization." In: *Graph Drawing and Network Visualization - 26th International Symposium, GD 2018, Barcelona, Spain, September 26-28, 2018, Proceedings*. Ed. by Biedl, T. and Kerren, A. Vol. 11282. Lecture Notes in Computer Science. Springer, 2018, pp. 286–299. DOI: 10.1007/978-3-030-04414-5_20.

[25] Diestel, R. *Graph Theory*. Springer-Verlag, Heidelberg, 2016.

[26] Doignon, J.-P., Ducamp, A., and Falmagne, J.-C. "On realizable biorders and the biorder dimension of a relation." In: *Journal of Mathematical Psychology* 28.1 (1984), pp. 73–109.

[27] Dua, D. and Graff, C. *UCI Machine Learning Repository*. 2017. URL: `https://archive.ics.uci.edu/ml`.

[28] Dürrschnabel, D., Hanika, T., and Stubbemann, M. "FCA2VEC: Embedding Techniques for Formal Concept Analysis." In: *Complex Data Analytics with Formal Concept Analysis*. Ed. by Missaoui, R., Kwuida, L., and Abdessalem, T. Springer International Publishing, 2022, pp. 47–74. DOI: 10.1007/978-3-030-93278-7_3.

[29] Dürrschnabel, D., Hanika, T., and Stumme, G. *Dataset of User Study for DimDraw*. Zenodo, Oct. 2020. DOI: 10.5281/zenodo.4075207.

[30] Dürrschnabel, D., Hanika, T., and Stumme, G. "DimDraw - A Novel Tool for Drawing Concept Lattices." In: *Supplementary Proceedings of ICFCA 2019 Conference and Workshops, Frankfurt, Germany, June 25-28, 2019*. Ed. by Cristea, D., Ber, F. L., Missaoui, R., Kwuida, L., and Sertkaya, B. Vol. 2378. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 60–64. URL: https://ceur-ws.org/Vol-2378/shortAT6.pdf.

[31] Dürrschnabel, D., Hanika, T., and Stumme, G. "Drawing Order Diagrams Through Two-Dimension Extension." In: *CoRR* abs/1906.06208 (2019). arXiv: 1906.06208. URL: http://arxiv.org/abs/1906.06208.

[32] Dürrschnabel, D., Hanika, T., and Stumme, G. *Experimental Evaluation of DimDraw*. June 2019. URL: https://doi.org/10.5281/zenodo.3242627.

[33] Dürrschnabel, D., Hanika, T., and Stumme, G. *User Study of DimDraw*. Zenodo, Oct. 2020. DOI: 10.5281/zenodo.4075215.

[34] Dürrschnabel, D., Koyda, M., and Stumme, G. "Attribute Selection Using Contranominal Scales." In: *Graph-Based Representation and Reasoning - 26th International Conference on Conceptual Structures, ICCS 2021, Virtual Event, September 20-22, 2021, Proceedings*. Ed. by Braun, T., Gehrke, M., Hanika, T., and Hernandez, N. Vol. 12879. Lecture Notes in Computer Science. Springer, 2021, pp. 127–141. DOI: 10.1007/978-3-030-86982-3_10.

[35] Dürrschnabel, D. and Stumme, G. *Code for Discovering Locally Maximal Bipartite Subgraphs*. [Online; accessed 14-February-2023]. 2023. URL: https://github.com/domduerr/bipartite.

[36] Dürrschnabel, D. and Stumme, G. "Force-Directed Layout of Order Diagrams Using Dimensional Reduction." In: *Formal Concept Analysis - 16th International Conference, ICFCA 2021, Strasbourg, France, June 29 - July 2, 2021, Proceedings*. Ed. by Braud, A., Buzmakov, A., Hanika, T., and Ber, F. L. Vol. 12733. Lecture Notes in Computer Science. Springer, 2021, pp. 224–240. DOI: 10.1007/978-3-030-77867-5_14.

[37] Dürrschnabel, D. and Stumme, G. *Force-Directed Layout of Order Diagrams using Dimensional Reduction - Source Code*. [Online; accessed 23-February-2023]. 2021. URL: https://github.com/domduerr/redraw.

[38] Dürrschnabel, D. and Stumme, G. "Greedy Discovery of Ordinal Factors." In: *CoRR* abs/2302.11554 (2023). arXiv: 2302.11554. URL: https://arxiv.org/abs/2302.11554.

[39] Dürrschnabel, D. and Stumme, G. *Greedy Ordinal Factor Analysis - Demonstration Platform*. [Online; accessed 23-February-2023]. 2022. URL: `https://factoranalysis.github.io/ordinal/`.

[40] Dürrschnabel, D. and Stumme, G. *Greedy Ordinal Factor Analysis - Source Code*. [Online; accessed 23-February-2023]. 2022. URL: `https://figshare.com/s/8edc72eaebc53e74f146`.

[41] Dürrschnabel, D. and Stumme, G. "Maximal Ordinal Two-Factorizations." In: *Accepted to: 28th International Conference on Conceptual Structures*.

[42] Dushnik, B. and Miller, E. W. "Partially ordered sets." In: *American journal of mathematics* 63.3 (1941), pp. 600–610.

[43] Eades, P. "A heuristic for graph drawing." In: *Congressus Numerantium* 42 (1984), pp. 149–160.

[44] Eén, N. and Sörensson, N. "An Extensible SAT-solver." In: *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*. Ed. by Giunchiglia, E. and Tacchella, A. Springer, 2003, pp. 502–518. DOI: `10.1007/978-3-540-24605-3\_37`.

[45] Espadoto, M., Martins, R. M., Kerren, A., Hirata, N. S. T., and Telea, A. C. "Toward a Quantitative Survey of Dimension Reduction Techniques." In: *IEEE Trans. Vis. Comput. Graph.* 27.3 (2021), pp. 2153–2173. DOI: `10.1109/TVCG.2019.2944182`.

[46] Euler, L. "Solutio problematis ad geometriam situs pertinentis." In: *Commentarii academiae scientiarum Petropolitanae* (1741), pp. 128–140.

[47] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., eds. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996. ISBN: 0-262-56097-6.

[48] Felsner, S. and Reuter, K. "The Linear Extension Diameter of a Poset." In: *SIAM J. Discret. Math.* 12.3 (1999), pp. 360–373. URL: `https://doi.org/10.1137/S0895480197326139`.

[49] Floyd, R. W. "Algorithm 97: Shortest path." In: *Commun. ACM* 5.6 (1962), p. 345. DOI: `10.1145/367766.368168`.

[50] Freese, R. "Automated Lattice Drawing." In: *Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings*. Ed. by Eklund, P. W. Vol. 2961. Lecture Notes in Computer Science. Springer, 2004, pp. 112–127. DOI: `10.1007/978-3-540-24651-0\_12`.

[51] Fruchterman, T. M. J. and Reingold, E. M. “Graph Drawing by Force-directed Placement.” In: *Softw. Pract. Exp.* 21.11 (1991), pp. 1129–1164. DOI: 10.1002/spe.4380211102.

[52] Ganguly, D., Roy, D., Mitra, M., and Jones, G. J. F. “Word Embedding based Generalized Language Model for Information Retrieval.” In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. Ed. by Baeza-Yates, R., Lalmas, M., Moffat, A., and Ribeiro-Neto, B. A. ACM, 2015, pp. 795–798. DOI: 10.1145/2766462.2767780.

[53] Ganter, B. “Conflict Avoidance in Additive Order Diagrams.” In: *J. Univers. Comput. Sci.* 10.8 (2004), pp. 955–966. DOI: 10.3217/jucs-010-08-0955.

[54] Ganter, B. *Diskrete Mathematik: Geordnete Mengen*. 1st ed. Springer-Lehrbuch. Springer Spektrum Berlin, Heidelberg, 2013. DOI: 10.1007/978-3-642-37500-2.

[55] Ganter, B. “Two Basic Algorithms in Concept Analysis.” In: *Formal Concept Analysis, 8th International Conference, ICFCA 2010, Agadir, Morocco, March 15-18, 2010. Proceedings*. Ed. by Kwuida, L. and Sertkaya, B. Vol. 5986. Lecture Notes in Computer Science. Springer, 2010, pp. 312–340. DOI: 10.1007/978-3-642-11928-6_22.

[56] Ganter, B. and Glodeanu, C. V. “Ordinal Factor Analysis.” In: *Formal Concept Analysis - 10th International Conference, ICFCA 2012, Leuven, Belgium, May 7-10, 2012. Proceedings*. Ed. by Domenach, F., Ignatov, D. I., and Poelmans, J. Vol. 7278. Lecture Notes in Computer Science. Springer, 2012, pp. 128–139. DOI: 10.1007/978-3-642-29892-9_15.

[57] Ganter, B. and Wille, R. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, 1999.

[58] Glodeanu, C. V. and Ganter, B. “Applications of Ordinal Factor Analysis.” In: *Formal Concept Analysis, 11th International Conference, ICFCA 2013, Dresden, Germany, May 21-24, 2013. Proceedings*. Ed. by Cellier, P., Distel, F., and Ganter, B. Vol. 7880. Lecture Notes in Computer Science. Springer, 2013, pp. 109–124. DOI: 10.1007/978-3-642-38317-5_7.

[59] Glodeanu, C. V. and Konecny, J. “Ordinal Factor Analysis of Graded Data.” In: *Formal Concept Analysis - 12th International Conference, ICFCA 2014, Cluj-Napoca, Romania, June 10-13, 2014. Proceedings*. Ed. by Glodeanu, C. V., Kaytoue, M., and Sacarea, C. Vol. 8478. Lecture Notes in Computer Science. Springer, 2014, pp. 128–140. DOI: 10.1007/978-3-319-07248-7_10.

[60] Golumbic, M. C. "The complexity of comparability graph recognition and coloring." In: *Computing* 18.3 (1977), pp. 199–208. DOI: 10.1007/BF02253207.

[61] Grover, A. and Leskovec, J. "node2vec: Scalable Feature Learning for Networks." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., and Rastogi, R. ACM, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754.

[62] Hanika, T. and Hirth, J. "Conexp-Clj - A Research Tool for FCA." In: *Supplementary Proceedings of ICFCA 2019 Conference and Workshops, Frankfurt, Germany, June 25-28, 2019*. Ed. by Cristea, D., Ber, F. L., Missaoui, R., Kwuida, L., and Sertkaya, B. Vol. 2378. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 70–75. URL: https://ceur-ws.org/Vol-2378/shortAT8.pdf.

[63] Hanika, T. and Hirth, J. "Knowledge cores in large formal contexts." In: *Ann. Math. Artif. Intell.* 90.6 (2022), pp. 537–567. DOI: 10.1007/s10472-022-09790-6.

[64] Hanika, T., Marx, M., and Stumme, G. "Discovering Implicational Knowledge in Wikidata." In: *Formal Concept Analysis - 15th International Conference, ICFCA 2019, Frankfurt, Germany, June 25-28, 2019, Proceedings*. Ed. by Cristea, D., Ber, F. L., and Sertkaya, B. Vol. 11511. Lecture Notes in Computer Science. Springer, 2019, pp. 315–323. DOI: 10.1007/978-3-030-21462-3_21.

[65] Ho, V. T., Stepanova, D., Gad-Elrab, M. H., Kharlamov, E., and Weikum, G. "Rule Learning from Knowledge Graphs Guided by Embedding Models." In: *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*. Ed. by Vrandecic, D., Bontcheva, K., Suárez-Figueroa, M. C., Presutti, V., Celino, I., Sabou, M., Kaffee, L., and Simperl, E. Vol. 11136. Lecture Notes in Computer Science. Springer, 2018, pp. 72–90. DOI: 10.1007/978-3-030-00671-6_5.

[66] Hoffman, P. E. and Grinstein, G. G. "A Survey of Visualizations for High-Dimensional Data Mining." In: *Information Visualization in Data Mining and Knowledge Discovery*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 47–82. ISBN: 1558606890.

[67] Hong, S., Eades, P., and Lee, S. H. "Drawing series parallel digraphs symmetrically." In: *Comput. Geom.* 17.3-4 (2000), pp. 165–188. DOI: 10.1016/S0925-7721(00)00020-1.

[68] Hopcroft, J. E. and Tarjan, R. E. "Efficient Planarity Testing." In: *J. ACM* 21.4 (1974), pp. 549–568. DOI: 10.1145/321850.321852.

[69] IMDb.com. *IMDb Datasets*. [Online; accessed 06-October-2021]. 2004. URL: https://www.imdb.com/interfaces/.

[70] Jain, P. M. and Shandliya, V. "A survey paper on comparative study between principal component analysis (PCA) and exploratory factor analysis (EFA)." In: *International Journal of Computer Science and Applications* 6.2 (2013), pp. 373–375.

[71] Karp, R. M. "Reducibility Among Combinatorial Problems." In: *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*. Ed. by Miller, R. E. and Thatcher, J. W. Plenum Press, New York, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

[72] Katoch, S., Chauhan, S. S., and Kumar, V. "A review on genetic algorithm: past, present, and future." In: *Multim. Tools Appl.* 80.5 (2021), pp. 8091–8126. DOI: 10.1007/s11042-020-10139-6.

[73] Keprt, A. "Algorithms for Binary Factor Analysis." PhD thesis. PhD thesis, 2006.

[74] Keprt, A. and Snásel, V. "Binary Factor Analysis with Help of Formal Concepts." In: *Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications, Ostrava, Czech Republic, September 23-24, 2004*. Ed. by Snásel, V. and Belohlávek, R. Vol. 110. CEUR Workshop Proceedings. CEUR-WS.org, 2004. URL: https://ceur-ws.org/Vol-110/paper10.pdf.

[75] Kersting, K., Peters, J., and Rothkopf, C. A. "Was ist eine Professur fuer Kuenstliche Intelligenz?" In: *CoRR* abs/1903.09516 (2019). arXiv: 1903.09516. URL: https://arxiv.org/abs/1903.09516.

[76] Koopmann, T., Stubbemann, M., Kapa, M., Paris, M., Buenstorf, G., Hanika, T., Hotho, A., Jäschke, R., and Stumme, G. "Proximity dimensions and the emergence of collaboration: a HypTrails study on German AI research." In: *Scientometrics* 126.12 (2021), pp. 9847–9868. DOI: 10.1007/s11192-021-03922-1.

[77] Kornaropoulos, E. M. and Tollis, I. G. “Weak Dominance Drawings and Linear Extension Diameter.” In: *CoRR* abs/1108.1439 (2011). arXiv: 1108.1439. URL: https://arxiv.org/abs/1108.1439.

[78] Kornaropoulos, E. M. and Tollis, I. G. “Weak Dominance Drawings for Directed Acyclic Graphs.” In: *Graph Drawing - 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers*. Ed. by Didimo, W. and Patrignani, M. Vol. 7704. Lecture Notes in Computer Science. Springer, 2012, pp. 559–560. DOI: 10.1007/978-3-642-36763-2_52.

[79] Koyda, M. and Stumme, G. “Boolean Substructures in Formal Concept Analysis.” In: *Formal Concept Analysis - 16th International Conference, ICFCA 2021, Strasbourg, France, June 29 - July 2, 2021, Proceedings*. Ed. by Braud, A., Buzmakov, A., Hanika, T., and Ber, F. L. Vol. 12733. Lecture Notes in Computer Science. Springer, 2021, pp. 38–53. DOI: 10.1007/978-3-030-77867-5_3.

[80] Krajca, P. “Improving the Performance of Lindig-Style Algorithms with Empty Intersections.” In: *Graph-Based Representation and Reasoning - 26th International Conference on Conceptual Structures, ICCS 2021, Virtual Event, September 20-22, 2021, Proceedings*. Ed. by Braun, T., Gehrke, M., Hanika, T., and Hernandez, N. Vol. 12879. Lecture Notes in Computer Science. Springer, 2021, pp. 91–104. DOI: 10.1007/978-3-030-86982-3_7.

[81] Kunegis, J. “KONECT: the Koblenz network collection.” In: *22nd International World Wide Web Conference, WWW ’13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*. Ed. by Carr, L., Laender, A. H. F., Lóscio, B. F., King, I., Fontoura, M., Vrandecic, D., Aroyo, L., Oliveira, J. P. M. de, Lima, F., and Wilde, E. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 1343–1350. DOI: 10.1145/2487788.2488173.

[82] Kuznetsov, S. O., Makhazhanov, N., and Ushakov, M. “On Neural Network Architecture Based on Concept Lattices.” In: *Foundations of Intelligent Systems - 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings*. Ed. by Kryszkiewicz, M., Appice, A., Slezak, D., Rybinski, H., Skowron, A., and Ras, Z. W. Vol. 10352. Lecture Notes in Computer Science. Springer, 2017, pp. 653–663. DOI: 10.1007/978-3-319-60438-1_64.

[83] Kuznetsov, S. O. and Obiedkov, S. A. “Comparing performance of algorithms for generating concept lattices.” In: *J. Exp. Theor. Artif. Intell.* 14.2-3 (2002), pp. 189–216. DOI: 10.1080/09528130210164170.

[84] LeCun, Y., Bengio, Y., and Hinton, G. E. "Deep learning." In: *Nat.* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[85] Lewis, J. M. and Yannakakis, M. "The Node-Deletion Problem for Hereditary Properties is NP-Complete." In: *J. Comput. Syst. Sci.* 20.2 (1980), pp. 219–230. DOI: 10.1016/0022-0000(80)90060-4.

[86] Lindig, C. "Fast concept analysis." In: *Working with Conceptual Structures-Contributions to ICCS* 2000 (2000), pp. 152–161.

[87] Lozin, V. V. "On maximum induced matchings in bipartite graphs." In: *Inf. Process. Lett.* 81.1 (2002), pp. 7–11. DOI: 10.1016/S0020-0190(01)00185-5.

[88] Lund, C. and Yannakakis, M. "The Approximation of Maximum Subgraph Problems." In: *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings*. Ed. by Lingas, A., Karlsson, R. G., and Carlsson, S. Vol. 700. Lecture Notes in Computer Science. Springer, 1993, pp. 40–51. DOI: 10.1007/3-540-56939-1_60.

[89] McConnell, R. M. and Spinrad, J. P. "Linear-Time Transitive Orientation." In: *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 5-7 January 1997, New Orleans, Louisiana, USA*. Ed. by Saks, M. E. ACM/SIAM, 1997, pp. 19–25. URL: https://dl.acm.org/citation.cfm?id=314161.314172.

[90] Mikolov, T., Chen, K., Corrado, G., and Dean, J. "Efficient Estimation of Word Representations in Vector Space." In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Bengio, Y. and LeCun, Y. 2013. URL: https://arxiv.org/abs/1301.3781.

[91] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. "Distributed Representations of Words and Phrases and their Compositionality." In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q. 2013, pp. 3111–3119. URL: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

[92] *Mushroom*. UCI Machine Learning Repository. 1987.

[93] Nielsen, F. Å. “Wembedder: Wikidata entity embedding web service.” In: *CoRR* abs/1710.04099 (2017). arXiv: 1710.04099. URL: https://arxiv.org/abs/1710.04099.

[94] Nishizeki, T. and Rahman, M. S. *Planar Graph Drawing*. Vol. 12. Lecture Notes Series on Computing. World Scientific, 2004. ISBN: 981-256-033-5. DOI: 10.1142/5648.

[95] Pavlopoulos, G. A., Kontou, P. I., Pavlopoulou, A., Bouyioukos, C., Markou, E., and Bagos, P. G. “Bipartite graphs in systems biology and medicine: a survey of methods and applications.” In: *GigaScience* 7.4 (2018).

[96] Pearson, K. “LIII. On lines and planes of closest fit to systems of points in space.” In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572.

[97] Ristoski, P., Rosati, J., Noia, T. D., Leone, R. D., and Paulheim, H. “RDF2Vec: RDF graph embeddings and their applications.” In: *Semantic Web* 10.4 (2019), pp. 721–752. DOI: 10.3233/SW-180317.

[98] Roy, B. “Transitivité et connexité.” In: *Comptes Rendus Hebdomadaires Des Seances De L Academie Des Sciences* 249.2 (1959), pp. 216–218.

[99] Rudolph, S. “Using FCA for Encoding Closure Operators into Neural Networks.” In: *Conceptual Structures: Knowledge Architectures for Smart Applications, 15th International Conference on Conceptual Structures, ICCS 2007, Sheffield, UK, July 22-27, 2007, Proceedings*. Ed. by Priss, U., Polovina, S., and Hill, R. Vol. 4604. Lecture Notes in Computer Science. Springer, 2007, pp. 321–332. DOI: 10.1007/978-3-540-73681-3_24.

[100] Scott, D. “Measurement structures and linear inequalities.” In: *Journal of Mathematical Psychology* 1.2 (1964), pp. 233–247. ISSN: 0022-2496.

[101] Seshapanpu, J. *Students Performance in Exams*. [Online; accessed 10-November-2022]. 2018. URL: https://www.kaggle.com/datasets/spscientist/students-performance-in-exams.

[102] Sinz, C. “Towards an Optimal CNF Encoding of Boolean Cardinality Constraints.” In: *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*. Ed. by Beek, P. van. Vol. 3709. Lecture Notes in Computer Science. Springer, 2005, pp. 827–831. DOI: 10.1007/11564751_73.

[103] Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., and Lakhal, L. “Computing iceberg concept lattices with Titanic.” In: *Data & Knowledge Engineering* 42.2 (2002), pp. 189–222. ISSN: 0169-023X. DOI: https://doi.org/10.1016/S0169-023X(02)00057-5.

[104] Sugiyama, K., Tagawa, S., and Toda, M. “Methods for Visual Understanding of Hierarchical System Structures.” In: *IEEE Trans. Syst. Man Cybern.* 11.2 (1981), pp. 109–125. DOI: 10.1109/TSMC.1981.4308636.

[105] Suman, B. and Kumar, P. “A survey of simulated annealing as a tool for single and multiobjective optimization.” In: *J. Oper. Res. Soc.* 57.10 (2006), pp. 1143–1160. DOI: 10.1057/palgrave.jors.2602068.

[106] Trotter, W. *Combinatorics and Partially Ordered Sets: Dimension Theory*. The Johns Hopkins University Press, 1992.

[107] Trotter, W. T. and Bogart, K. P. “On the complexity of posets.” In: *Discret. Math.* 16.1 (1976), pp. 71–82. DOI: 10.1016/0012-365X(76)90095-9.

[108] Trukhanov, S., Balasubramaniam, C., Balasundaram, B., and Butenko, S. “Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations.” In: *Comput. Optim. Appl.* 56.1 (2013), pp. 113–130. DOI: 10.1007/s10589-013-9548-5.

[109] Wang, Z., Zhang, J., Feng, J., and Chen, Z. “Knowledge Graph Embedding by Translating on Hyperplanes.” In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. Ed. by Brodley, C. E. and Stone, P. AAAI Press, 2014, pp. 1112–1119. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[110] Warshall, S. “A Theorem on Boolean Matrices.” In: *J. ACM* 9.1 (1962), pp. 11–12. DOI: 10.1145/321105.321107.

[111] Wikipedia contributors. *Seven Bridges of Königsberg — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-February-2023]. 2023. URL: https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg.

[112] Wille, R. “Lattices in data analysis: how to draw them with a computer.” In: *Algorithms and order*. Springer, 1989, pp. 33–58.

[113] Wille, R. “Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts.” In: *Ordered Sets*. Ed. by Rival, I. Dordrecht: Springer Netherlands, 1982, pp. 445–470. ISBN: 978-94-009-7798-3.

[114] Wille, R. “Truncated Distributive Lattices: Conceptual Structures of Simple-Implicational Theories.” In: *Order* 20.3 (2003), pp. 229–238. DOI: 10.1023/B:ORDE.0000026494.22248.85.

[115] Wille, U. “Representation of Finite Ordinal Data in Real Vector Spaces.” In: *Data Analysis and Information Systems*. Ed. by Bock, H.-H. and Polasek, W. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 228–240. ISBN: 978-3-642-80098-6.

[116] Wu, Q. and Hao, J. "A review on algorithms for maximum clique problems." In: *Eur. J. Oper. Res.* 242.3 (2015), pp. 693–709. DOI: 10.1016/j.ejor.2014.09.064.

[117] Xiao, M. and Tan, H. "Exact algorithms for Maximum Induced Matching." In: *Inf. Comput.* 256 (2017), pp. 196–211. DOI: 10.1016/j.ic.2017.07.006.

[118] Yannakakis, M. "Edge-Deletion Problems." In: *SIAM J. Comput.* 10.2 (1981), pp. 297–309. DOI: 10.1137/0210021.

[119] Yannakakis, M. "The complexity of the partial order dimension problem." In: *SIAM Journal on Algebraic Discrete Methods* 3.3 (1982), pp. 351–358.

[120] Yevtushenko, S. A. "Computing and visualizing concept lattices." PhD thesis. Darmstadt University of Technology, Germany, 2004. URL: https://elib.tu-darmstadt.de/diss/000488.

[121] Zhang, Z., Li, T., Ding, C. H. Q., and Zhang, X. "Binary Matrix Factorization with Applications." In: *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*. IEEE Computer Society, 2007, pp. 391–400. DOI: 10.1109/ICDM.2007.99.

[122] *Zoo*. UCI Machine Learning Repository. 1990.

[123] Zschalig, C. "An FDP-Algorithm for Drawing Lattices." In: *Proceedings of the Fifth International Conference on Concept Lattices and Their Applications, CLA 2007, Montpellier, France, October 24-26, 2007*. Ed. by Eklund, P. W., Diatta, J., and Liquiere, M. Vol. 331. CEUR Workshop Proceedings. CEUR-WS.org, 2007. URL: https://ceur-ws.org/Vol-331/Zschalig.pdf.