FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Interactive Visual Exploration Interface for Large Bipartite Networks

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Daniel Steinböck, BSc
Matrikelnummer 00826088

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Univ.Ass. Dr.techn. Manuela Waldner, MSc

Wien, 30. Mai 2018

Daniel Steinböck	Eduard Gröller

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Interactive Visual Exploration Interface for Large Bipartite Networks

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Daniel Steinböck, BSc
Registration Number 00826088

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Univ.Ass. Dr.techn. Manuela Waldner, MSc

Vienna, 30th May, 2018

Daniel Steinböck | Eduard Gröller

Erklärung zur Verfassung der Arbeit

Daniel Steinböck, BSc
Wehlistraße 131-143/18/5, 1020 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. Mai 2018

Daniel Steinböck

Kurzfassung

In dieser Arbeit präsentieren wir *BiCFlows*, einen neuartigen Ansatz zur interaktiven Exploration großer bipartiter Graphen. Die Entwicklung wurde durch die *Medientransparenz-Datenbank* angeregt. Diese öffentlich zugängliche Datenbank wurde von der österreichischen Bundesregierung eingerichtet, um Informationen über staatliche Werbe- und Subventionsausgaben bereitzustellen. Diese Datenbank weist dabei die Eigenschaften eines großen, gewichteten bipartiten Graphen auf.

Aktuelle Ansätze, die sich mit der Visualisierung der Medientransparenz-Datenbank befassen, sind dadurch limitiert, dass sie keinen ausreichenden Überblick über den gesamten Datensatz bieten. Andere Ansätze, die nicht speziell für die Medientransparenz-Datenbank entwickelt wurden, sich jedoch mit der Visualisierung von bipartiten Graphen befassen, sind zusätzlich durch ihre mangelnde Skalierbarkeit bei großen Datensätze begrenzt.

Aggregation ist ein häufig verwendetes Konzept, um die Datenmenge durch Gruppieren ähnlicher Datenobjekte zu reduzieren. Dies funktioniert nur, wenn die entsprechenden Eigenschaften in den Daten vorhanden sind, um sie für die Aggregation zu verwenden. Wenn diese zusätzlichen Informationen, wie in der Medientransparenz-Datenbank fehlen, müssen andere Aggregationstechniken verwendet werden. Da wir uns in unserem Ansatz mit bipartiten Graphen beschäftigen, verwenden wir das Konzept des Biclusterns um eine hierarchische Struktur innerhalb der Daten zu erstellen, die vom Benutzer interaktiv erkundet werden kann.

Wir haben gezeigt, dass BiCFlows nicht nur für die Medientransparenz-Datenbank, sondern auch für andere Datensätze verwendet werden kann, die die Eigenschaften eines gewichteten bipartiten Graphen aufweisen. Darüber haben wir eine Benutzerstudie durchgeführt, um BiCFlows mit bestehenden Konzepten zu vergleichen und Vor- und Nachteile zu diskutieren. Wir haben gezeigt, dass BiCFlows die Anwender in ihrem Explorationsprozess unterstützt und ihnen mehr Einblicke als mit bestehenden Ansätzen ermöglicht.

Abstract

In this thesis we introduce *BiCFlows*, a novel interactive visualization approach to explore large bipartite graphs. We were motivated by the *Media Transparency Database*, a public database established by the Austrian government to provide information about governmental advertising and subsidies expenses, which holds the characteristics of a large, weighted bipartite graph.

Current approaches that deal with the visualization of the Media Transparency Database are limited by the fact that they do not offer a sufficient overview of the whole dataset. Other existing approaches that are not particularly designed for the Media Transparency Database, but deal with the visualization of bipartite graphs are in addition limited by their lack of scalability for large datasets.

Aggregation is an often used concept in reducing the amount of data by grouping together similar data objects. This only works if the appropriate object properties are present in the data to use them for the aggregation. If this additional information is missing, like in the Media Transparency Database, other aggregation techniques have to be used. Since we are dealing with bipartite graphs in our approach, we use the concept of biclustering to establish a hierarchical structure within the data that can be interactively explored by the user.

We showed that BiCFlows cannot only be used for the Media Transparency Database, but also for other datasets that share the characteristics of a weighted bipartite graph. Furthermore, we conducted an insight-based user study to compare BiCFlows with existing concepts and discussed advantages and drawbacks. We showed that BiCFlows supported users in their exploration process and let them gain more insight than with existing approaches.

Contents

CHAPTER 1

Introduction

1.1 Motivation

To offer more transparency in dealing with tax money and the possible influence of press opinion through media advertisement, the Austrian government passed a law concerning this matter in 2011. The law pledged public authorities and other organizations supervised by the Autrian Court of Audit to disclose their press subsidies and advertising objectives [Rec17]. This so-called *Medienkooperations- und -förderungs-Transparenzgesetz* [Bun17] came into force in July 2012 and henceforth obligated these organizations to quarterly report their expenses, exceeding € 5000, to the *Kommunikationsbehörde Austria (KommAustria)*. There are three paragraphs that form the legal basis for the obligation of disclosure:

- **§2:** Advertising objectives,
- **§4:** press subsidies, and
- **§31:** program fees of the Austrian Broadcasting Corporation (ORF).

The data is quarterly published at the website [uTRG17] of the *Rundfunk- und Telekom-Regulierungs-GmbH*, the office of KommAustria. There, the colloquially called *Media Transparency Database* is organized as shown in Figure 1.1. The properties of this database, which in fact holds the characteristics of a bipartite graph, will be covered in detail in Section 2.2.

This database can be used by journalists to reveal interesting relationships between public and media organizations, by media organizations themselves to see if they are disadvantaged in comparison to their competitors concerning advertises or funding, and by the general public, to find out where and how their taxes are spent. There are several

Rechtsträger	Quartal	Bekanntgabe	Medium	Euro
Stadt Wien	20123	2	Kronen Zeitung	610219.45
Bundeskanzleramt	20162	4	Kroatischer Presseverein	80000.00
VERBUND AG	20164	2	Österreich	33106.75

Figure 1.1: Description of columns (left to right) in the Media Transparency Database: legal entity, year with trailing quarter, legal basis of publication (§2=advertising, §4=subsidies), media organization, sum.

points that are of interest for journalists and for their readers. One of the primary interest lies in the expenses of the current federal government [der18]. This includes, besides the federal chancellery, all federal ministries. Moreover, it would be of interest if certain ministries advertise in the same media and also if there is a connection between ministries administrated by ministers of the same political party. Following ministry expenses over time could be problematic, since the names of most of the ministries change with every legislative period. This happens because certain departments are merged or split. Besides ministries, the expenses of all nine federal states can be of interest, again, also to determine if there are differences between politically different ministered states. These political differences can also be investigated during election years. Depending on the type of election, it can be interesting which ministry or state advertises in which media organization the most.

According to a domain expert we interviewed, it is very difficult to determine the full amount of advertising objects or press subsidies for certain media organizations. The reason for this lies in their corporate structure. There are media organizations, like *Mediengruppe Österreich GmbH*, which not only comprises the daily newspaper *Österreich*, but also *Madonna*, *wetter.at*, *oe24.at*, and more than 50 other media organizations. These organizations are all listed separately in the Media Transparency Database, making it difficult to connect them or to understand them as a whole.

Until 2017, around 1200 legal entities reported their advertising spendings to over 4200 media companies for the past 18 quarters, resulting in a database with more than 34000 entries that will continue to grow every quarter. Moreover, these cash flows are not normally distributed (see Figure 1.2). There is only a minority of organizations that exchanged a large amount of money. The vast majority share smaller sums. This characteristic will play an important role when we talk about the design choices in Chapter 4.

1.2 Problem Statement

Currently, there are a few websites that deal with the visualization of the Media Transparency Database, but each one has their own drawbacks. *Paroli-Magazin* [Mag17] published a couple of visualizations (see Figure 1.3), but their interactivity is very restricted. They only list governmental departments and federal states as legal entities

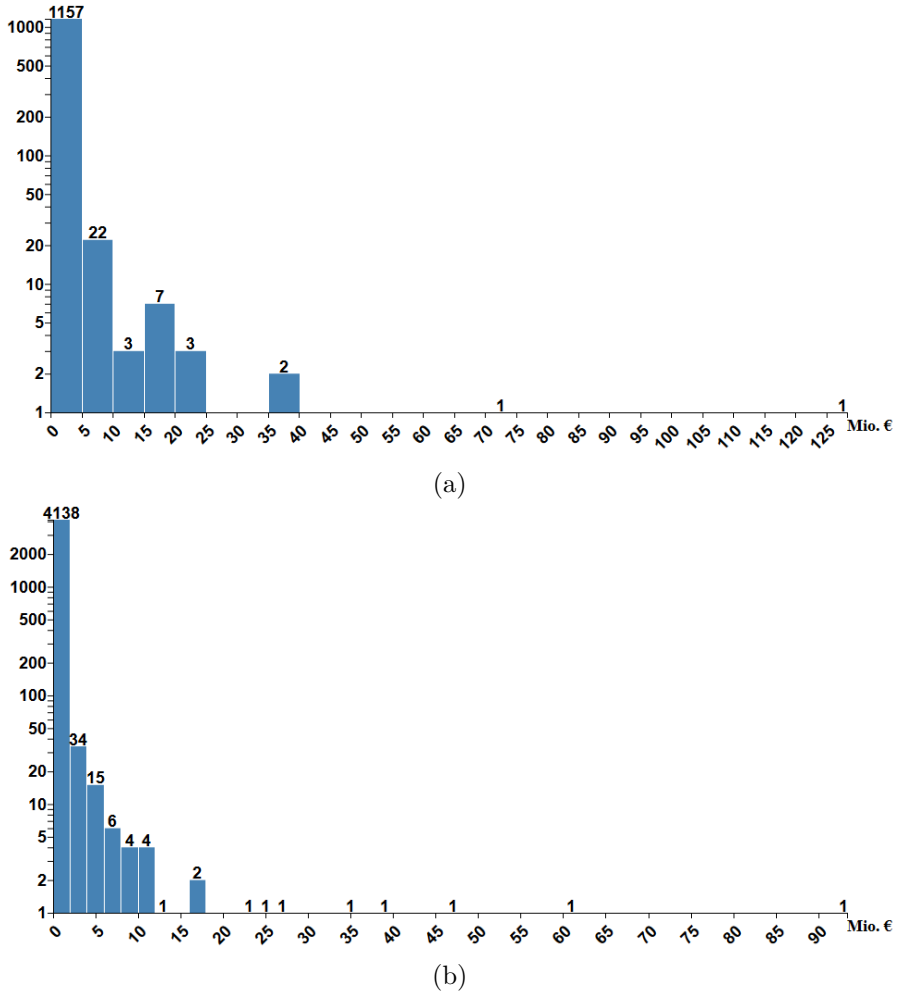


Figure 1.2: Histograms showing the distribution of aggregated cash flows over all years of: (a) legal entities and (b) media organizations.

and moreover, there is no way to distinguish between advertising and subsidies. The dashboard visualization by Rind et al. [RPNA16], which is also available online [Pfa17], offers different linked views and the possibility to filter the data (see Figure 1.4). However, this visualization lacks a general overview of the data, since it only shows the ten legal entities and media organizations associated with the highest cash flows, respectively. The web-based tool by Salhofer et al. [Sal17] of FH Joanneum also offers interactive exploration of the data and provides different visualizations (see Figure 1.5), but they are spread over several websites, which makes it difficult for the user to combine the gathered information from the different visualizations. What all of these visualizations have in common is that they lack some sort of overview that gives a broad survey of the interconnections between the different governmental and media organizations. Although the visualization in the top left corner of Figure 1.5 shows a geographical overview and

Figure 1.3b shows aggregated daily newspapers, the corresponding properties are not available in the raw data of the Media Transparency Database. Hence, these assignments had to be done manually.

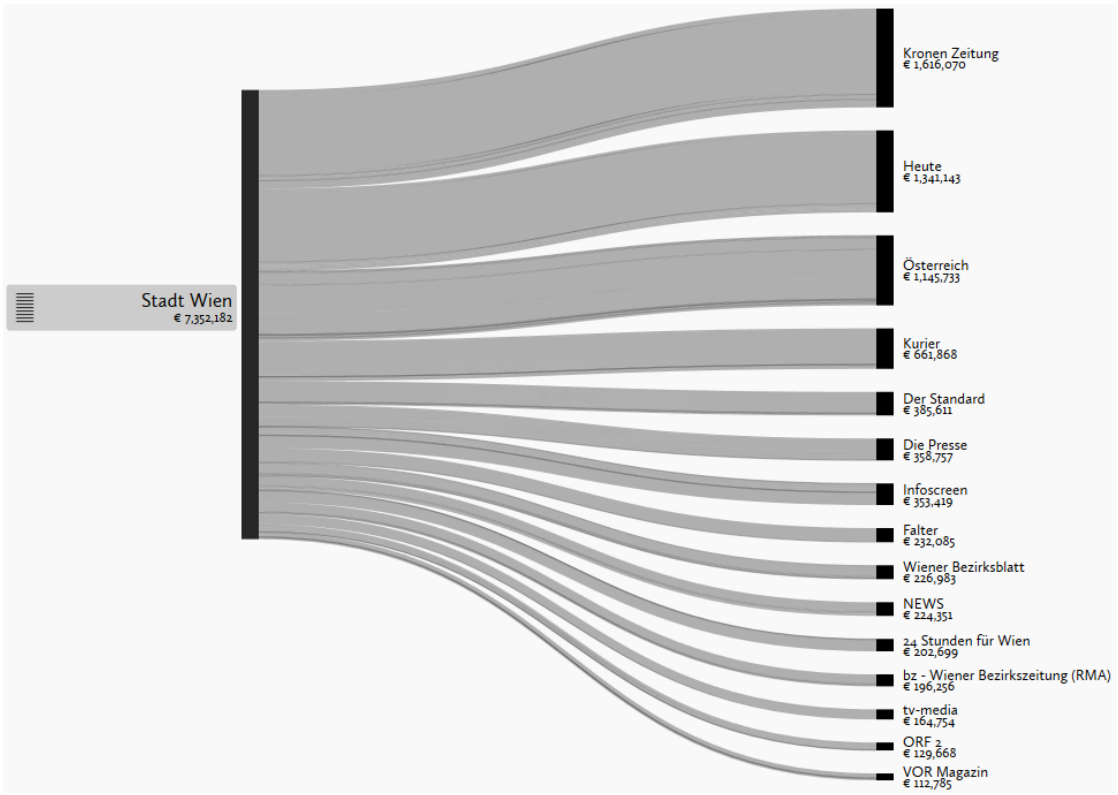
1.3 Contribution

Since there are several thousands of data objects, they cannot be visualized in classical diagrams and therefore have to be aggregated somehow to reduce the number of depicted entities. The aggregation cannot be applied by geographical regions or other hierarchies due to the fact that these additional categories are not available in the dataset. Thus, there are two possible approaches: a Cut-Off approach, where only a few single data objects are shown and the rest gets aggregated to one single new data object or a clustering approach, where the data objects are iteratively grouped based on their similarity. Clustering is a commonly used aggregation technique to establish a better overview over large data [EF10]. In our case, we focus on how bipartite data with no inherent hierarchical structure can be explored using the concept of biclustering.

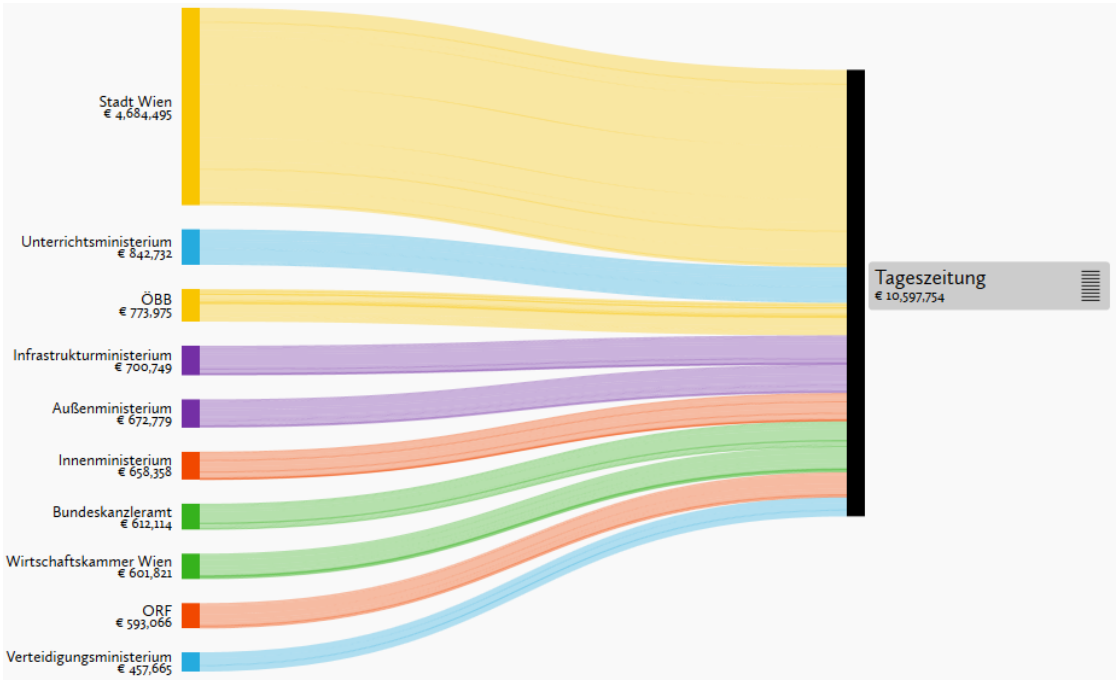
In this thesis we therefore present *BiCFlows*, a novel visual exploration interface for large bipartite graphs. The difference to already existing approaches, like the ones covered in Section 1.2, is that our exploration interface scales with the size of the underlying dataset. We kept the aspect of a web-based visualization, since it offers not only better accessibility, but also helps to separate computationally expensive operations of data processing on the server from the user interface on the client side. Besides this, offering a software that has to be installed on the user's computer runs into the problem with having to handle different operating systems. Even if the software is developed in an operating system independent programming language, it will be still too cumbersome for technically unversed users to install it. To examine assets and drawbacks of our approach, we conducted a user study where we compared BiCFlows with the concept of traditional Cut-Off approaches.

The contributions of this thesis are summarized in the following list:

- A visual encoding that is able to visualize weighted bipartite graphs with several thousands of nodes and edges by using the concept of biclustering (see Section 2.3).
- The technical infrastructure to visualize and interactively explore this huge amount of data in a web-client (see Section 4 and 5).
- A formal insight-based evaluation, where the advantages and disadvantages of BiCFlows compared to unclustered approaches were examined (see Chapter 7).



(a)



(b)

Figure 1.3: Two examples of visualizations at the website of *Paroli-Magazin* [Mag17]: (a) expenses of *Stadt Wien* and (b) expenses of ministries for daily newspapers.

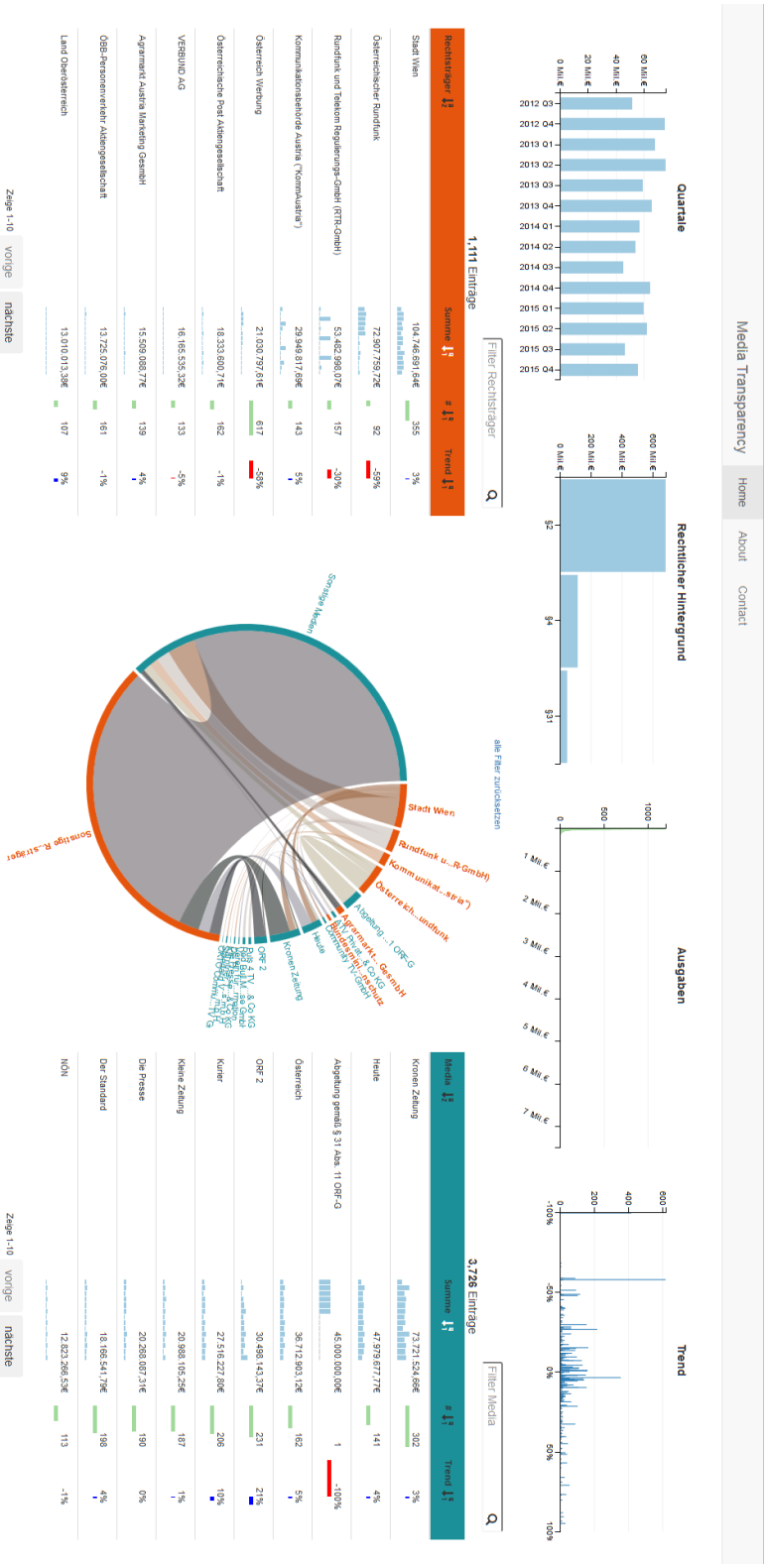


Figure 1.4: Dashboard visualization by Rind et al. [RPNA16].

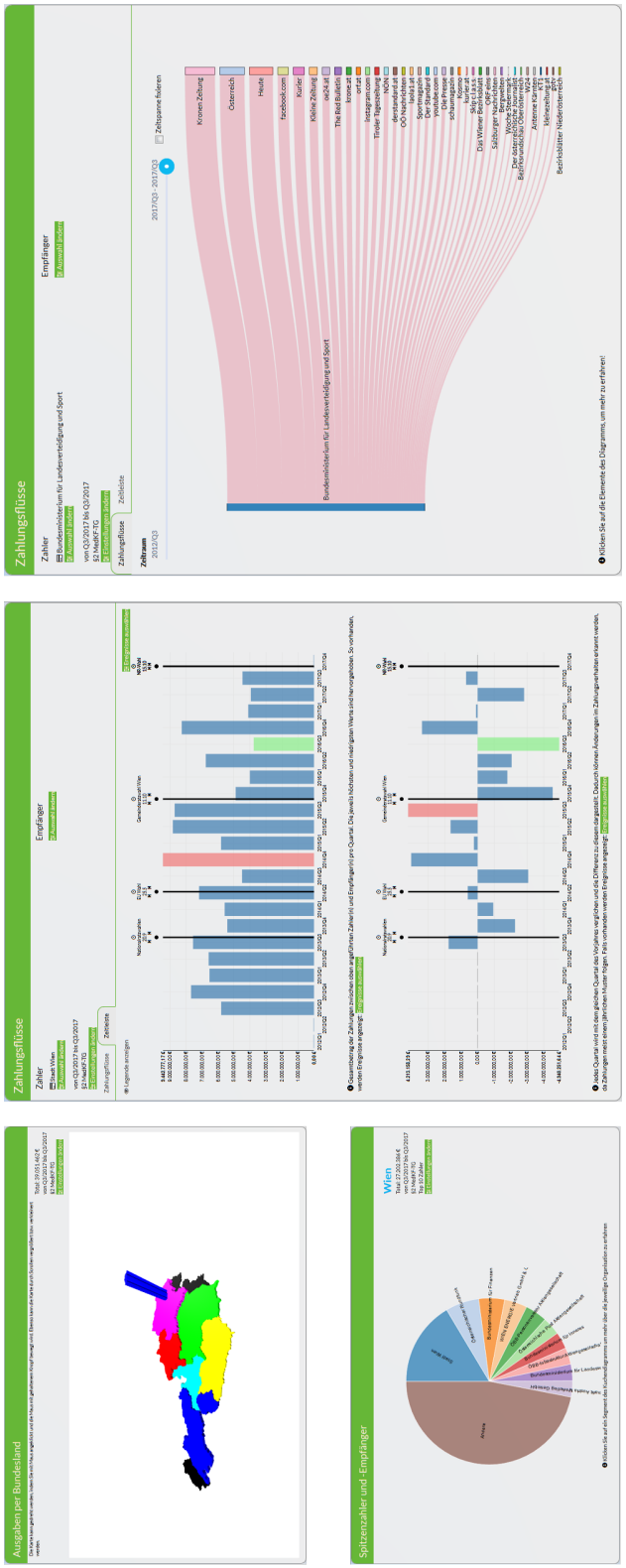


Figure 1.5: Different visualization offered at the website of Salhofer et al. [Sal17].

CHAPTER 2

Data Characteristics

This chapter aims to give some background information on the data characteristics and the techniques used in the implemented visualization tool. First, the definition of a graph and its properties are covered. Then, the special type of bipartite graph and its connection to the Media Transparency Database and similar datasets is discussed, after which the concept of *Biclustering* is explained.

2.1 Graphs

Graphs are widely used to depict connections or relationships between elements in different disciplines, such as biology, transportation, sociological, sociology or physics.

A definition for graphs is given by Gross et al. [GYZ13] as:

Definition. *A graph $G = (V, E)$ consists of two sets V and E, where the elements of V are called nodes and the elements of E are called edges. Each edge has a set of two nodes associated to it, which are called its endpoints. An edge is said to join its endpoints.*

Graphs can be divided into two classes: directed and undirected ones. While in directed graphs, edges are ordered pairs of vertices $E(v_0, v_1)$ and determine some sort of flow, in undirected graphs the pairs of vertices are unordered. Every edge can also have a weight ω. Directed, weighted graphs are sometimes referred to as networks [vLKS+11].

Every graph can be further distinguished by its properties. These properties include the number of nodes $|V|$, the number of edges $|E|$, density D or connectivity κ. The density is a ratio between number of edges and number of vertices and is defined as $D = \frac{2|E|}{|V|(|V|-1)}$. In dense graphs the number of edges is close to $|V|^2$, hence $D \simeq 1$, whereas sparse graphs have a much smaller number of edges than $|V|^2$. The connectivity of a graph is defined as the minimum number of vertices that must be removed to get

a disconnected graph [Rah17]. Another property of a graph is its completeness, where a graph is called complete if every node in the graph is connected to every other node through an edge. Furthermore, a clique is a complete subgraph of any arbitrary graph.

When talking about *large* graphs, there are different definitions. Graphs can be described as large with 100 or with 1 million nodes. Although there is no exact definition of large graphs, the number of edges, density, and connectivity contribute to it as well [vLKS+11].

2.2 Bipartite Graph

If a graph can be partitioned into k different set of nodes, which are independent and disjoint, the graph is called *k-partite* graph. A special case where $k = 2$ is called a bipartite graph.

The definition of a bipartite graph is given by Rahman [Rah17] as:

Definition. *Let $G = (V, E)$ be a graph. G is called a bipartite graph if the vertex set V of G can be partitioned into two disjoint nonempty sets V_1 and V_2, both of which are independent. A subset of vertices $V' \subseteq V$ is called an independent set in G if for every pair of vertices $u, v \in V'$, there is no edge in G joining the two vertices u and v. Each edge of a bipartite graph G thus joins exactly one vertex of V_1 to exactly one vertex of V_2.*

Moreover, in a weighted bipartite graph, every edge connecting a vertex of V_1 with a vertex of V_2 has a weight $\omega \geq 0$. Figure 2.1 shows an example of a bipartite graph and its two independent, disjoint sets of vertices.

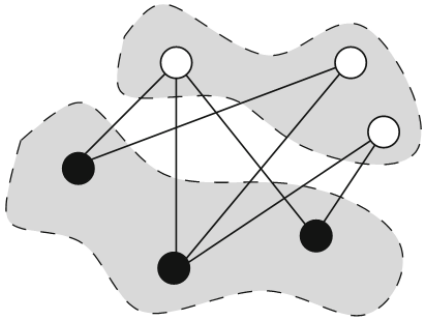


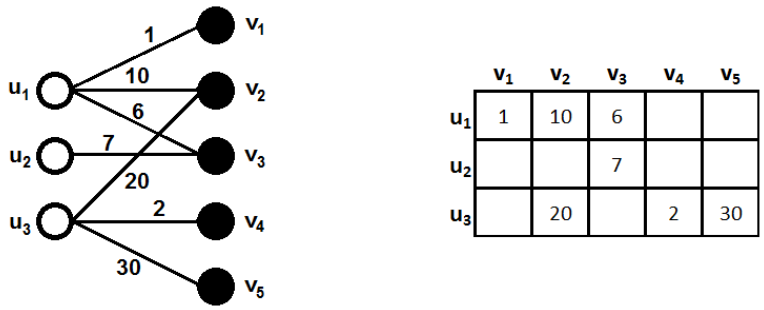
Figure 2.1: Example of a bipartite graph and its two independent sets of nodes (white and black) [Rah17].

As mentioned in Section 1.1, the Media Transparency Database holds the characteristics of a weighted bipartite graph. In our case, the two independent sets of nodes are on the one hand the legal entities and on the other hand the media organizations. The edges connecting nodes from one set with nodes from the other one represent the cash flows between legal entities and media organization, where the sum of money acts as edge weight. Other examples, that are covered in detail in Chapter 6, are publication and

movie data. In publication data, the two sets of nodes are authors and keywords where the edges between them represent how often authors use certain keywords. The movie data consists of movies and viewers where their interconnection is represented by the sum a viewer spent for a certain movie.

2.3 Biclustering

To offer a better overview of large bipartite graphs, where many data objects are present, some sort of aggregations has to be applied. In this context, it is important to understand that a bipartite graph can also be illustrated as a weighted biadjacency matrix, where the rows represent nodes of one set, the columns nodes of the other set and each matrix element the corresponding edge weight between two nodes. Figure 2.2 shows an example of such a matrix.



	v_1	v_2	v_3	v_4	v_5
u_1	1	10	6		
u_2			7		
u_3		20		2	30

Figure 2.2: Example of a weighted bipartite graph with corresponding biadjacency matrix.

The idea is now to find groups of rows and columns where their corresponding elements are most similar [EB15]. Finding and determining such groups is called *Clustering* and is a well known data mining technique used in many different scientific disciplines over the last decades [Ber06, Jai10]. In case of bipartite graphs, an approach called *Biclustering* [Mir98] or *Co-Clustering* [Dhi01] has been developed. It has been extensively studied over the last couple of years, mostly in Bioinformatics for analyzing gene expression data [MO04, JTZ04, PGAR15] and in document classification [Dhi01, RDF06, Bic10].

The concept of biclustering is that it simultaneously rearranges rows and columns of a biadjacency matrix to form clusters of certain (dis)similarity. These clusters are called biclusters. Madeira and Oliveira [MO04] identified four different types of biclusters that can be found, in terms of similarity:

1. Constant values, where all values within a bicluster are the same.

2. Constant values on rows or columns, where all values within a row or column are the same.

3. Coherent values, where rows and columns can be obtained by adding or multiplying each with a constant value.

4. Coherent evolutions, where not the values, but the additive or multiplicative behavior within a bicluster is the same.

Figure 2.3a-e show examples of type 1-3, where the numeric values are directly used to define similarity. Figure 2.3f-j are examples of type 4, where not the numeric value, but the coherent behavior is considered.

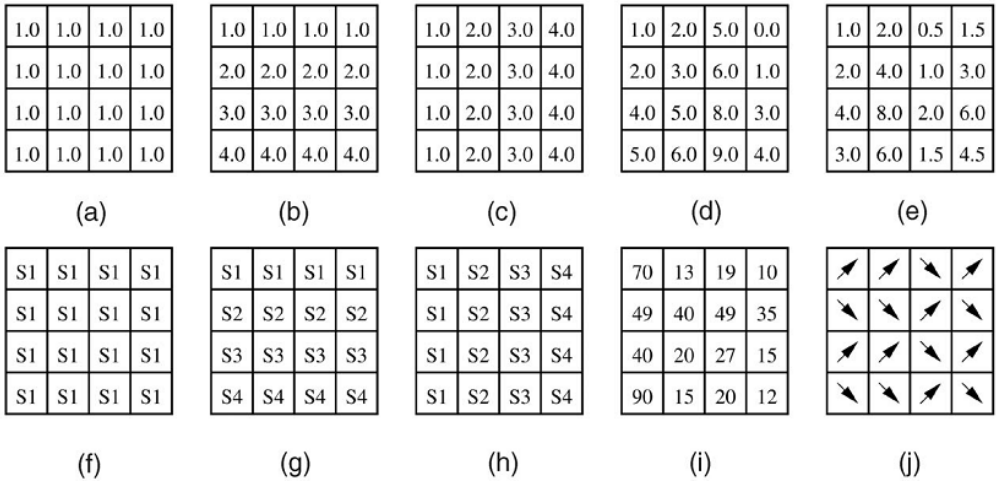


Figure 2.3: Different types of biclusters: (a) constant bicluster, (b) constant rows, (c) constant columns, (d) coherent values (additive), (e) coherent values (multiplicative), (f) overall coherent evolution, (g) coherent evolution on the rows, (h) coherent evolution on the columns, (i) coherent evolution on the columns, (j) coherent sign changes on rows and columns [MO04].

During the process of finding similar clusters, biclustering algorithms assume a specific structure of the underlying data matrix. Figure 2.4 shows some examples of these structures. The most commonly assumed structures are the *block diagonal structure* (see Figure 2.4b), where every row and column are assigned to exactly one cluster and the *checkerboard structure* (see Figure 2.4c), where every row and column are assigned to multiple clusters.

Since biclustering is an NP-hard problem [TSS02], many different algorithms have been developed that try to improve the clustering process by optimizing the search heuristics. Pontes et al. [PGAR15] categorized two different types of biclustering algorithms: algorithms based on evaluation measures and non metric-based algorithms. The former use a quality measure during the search process to further improve the clustering. Different quality measures for biclustering have been proposed over the time, such as variance [Har72], *Mean Squared Residue* [CC00] or *relevance index* [YCN04]. Non metric-based

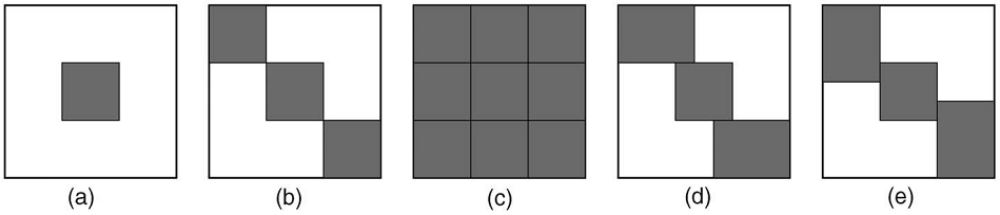


Figure 2.4: Different bicluster structures: (a) single bicluster, (b) exclusive row and column biclusters (block diagonal), (c) checkerboard structure, (d) exclusive rows biclusters, (e) exclusive columns biclusters [MO04].

algorithms do not use evaluation measures within their search for biclusters, but use different techniques that are, e.g., based on probabilistic models or linear mappings between vector spaces to determine the optimal clusters. Examples for this are the *Bayesian Biclustering model* developed by Gu and Liu [GL08] or the use of *Singular Value Decomposition* by Kluger et al. [KBCG03].

One evaluation-based algorithm not discussed by Pontes et al. [PGAR15] is of high relevance for this thesis. This approach is based on the *modularity measure*, originally introduced by Newman et al. [NG04, New06, WS05] as a measure for graph partitioning. *Modularity* is a metric that gives information about how densely nodes are connected together in a partition compared to the rest of the network. This measure has lately been adopted and proposed for biclustering by Labiod and Nadif [LN11]. An improved variation of their algorithm by Ailem et al. [ARN15], based on direct maximization of bipartite modularity, is used in this thesis. An important choice for using this algorithm is that it can also handle weighted biadjacency matrices, since the Media Transparency Database has this property. Most of the other proposed algorithms can only handle binary matrices. Although every weighted biadjacency matrix can be transformed into a binary biadjacency matrix, the weight property would be lost and would not be considered for the calculation of the modularity. As a consequence, this would distort the clustering result.

The idea behind this algorithm is that it iteratively tries to maximize the graph modularity for a predefined number of clusters. It starts with a random partitioning of the biadjacency matrices. This means that in the initial state, the matrix will be clustered randomly with the given number of clusters. Modularity is calculated for this partitioning and then iteratively optimized. Ailem et al. [ARN15] empirically evaluated that around 15 iterations are enough for their algorithm to determine the best partition. They compared it to other biclustering algorithms with different datasets containing binary and non-binary data, and showed that their approach outperforms all other algorithms. As metrics they used the normalized mutual information as well as the accuracy, where they determined how many data points were clustered correctly. They also made their algorithm publicly available as *Python* package, which we used in our server-side implementation (see Section 5.1).

CHAPTER 3

Related Work

While the previous chapter already covered the concept of biclustering and proposed biclustering algorithms, this chapter mainly focuses on related work regarding visualizations of graphs, especially bipartite graphs and biclustered data.

3.1 Visualizations of Graphs

One of the main reasons to visualize graphs is to explore and analyze the relationships of its nodes. There are two types of graphs that can be distinguished in regards of their time dependency. *Static* graphs are independent of time, whereas *dynamic* graphs can change over time. The change can involve their overall structure as well as their node or edge attributes. Beck et al. [BBD09] defined certain aesthetic criteria for dynamic graphs that can also be applied to static ones.

One of the main criteria is the visual readability, meaning that visual clutter through overlapping nodes, edges or other elements used in the visualization of the graph should be reduced. The graph should also use the space in which it is drawn into efficiently in the sense of keeping the graph in a compact form. Another criterion is the scalability of the visualization. Even if the number of vertices or edges increases, the visualization should still retain its readability.

In graph visualization, the most often used techniques are node-link diagrams and matrix-based representations. Node-link diagrams, the typical representation of graphs where nodes are connected through links, have been studied for various types of connections and diagram layouts [HMM00]. Von Landesberger et al. [vLKS+11] classified the following types of layouts based on their node placement:

- **Force-directed layouts:** Here, physical forces are simulated between nodes and edges. While attractive forces get assigned to endpoints of edges, repulsive forces are used between nodes.

- **Constraint-based layouts:** These layouts constrain the position of nodes. These constraints can include the alignment, the direction of edges, or the distance between nodes. An example can be seen in Figure 3.1, where the orthogonal layout consists only of straight vertical and horizontal edges.

- **Hierarchical layouts:** In this type of layout, the graph is divided into layers. The nodes are then, e.g., placed on these horizontal parallelly aligned layers.

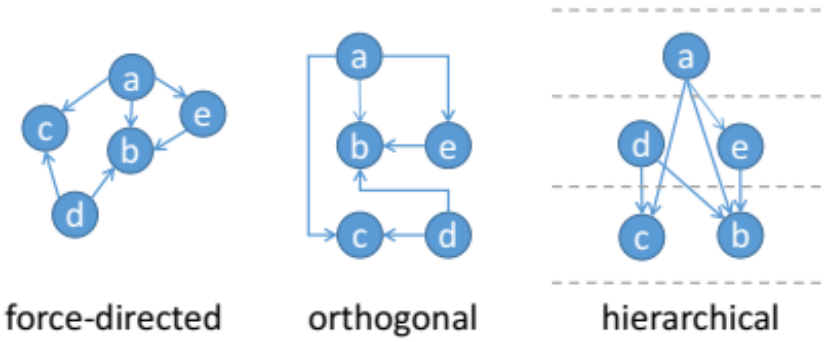


Figure 3.1: Three different layout types of node-link diagrams [BBDW17].

In matrix-based visualizations, nodes are represented as row and column keys of a matrix and a matrix cell depicts the interconnection between them (see Figure 3.2). If the graph is weighted, the matrix cells can also encode edge weights, either numerical or by color. Moreover, if the underlying graph is undirected, the matrix will be symmetrical. For bipartite graphs, the row and column keys will correspond to the nodes of the two independent sets, respectively (see Section 2.2).

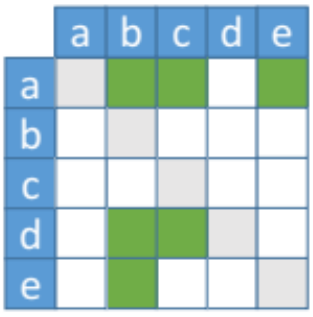


Figure 3.2: Example of a matrix-based layout [BBDW17].

Dependent on the type of data, node-link diagrams and matrix-based approaches have their own advantages and drawbacks. According to Ghoniem et al. [GFC04] node-link diagrams benefit from their intuitiveness and flexibility and are well suited for smaller and sparse datasets. Matrix-based visualizations profit from their better readability by eliminating occlusion problems and are thus more suitable for larger and dense datasets. However, there are also ideas of combining these two representations. Henry et al. [HFM07] proposed the concept of *NodeTrix*, which is a hybrid approach of node-link diagrams and matrix-based visualizations. It was developed to analyze and explore *small-world networks* [WS98] that are locally dense but globally sparse. The network is visualized as node-link diagram, where communities (densely connected nodes) are represented as adjacency matrices (see Figure 3.3). This concept allows one to explore relations within communities without visual clutter and also to follow connections between different communities.

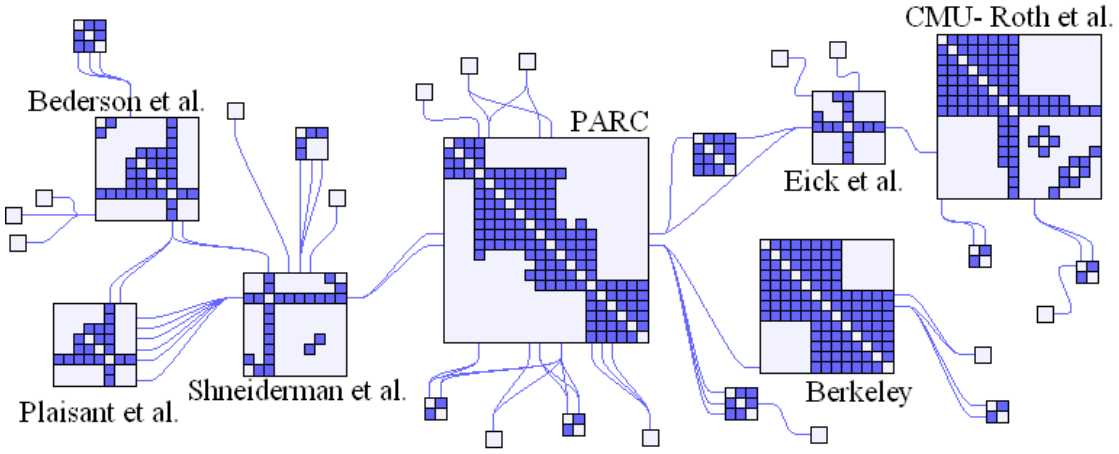


Figure 3.3: Example of *NodeTrix* showing a network of InfoVis co-authorship [HFM07].

3.2 Visualizations of Bipartite Graphs

As mentioned in Section 2.2, the Media Transparency Database can be seen as a bipartite graph, but has too many nodes and edges to display them all at once. Several approaches have been published that deal with an improved visualization of bipartite graphs. Misue et al. [Mis06] use a node-link diagram and suggest that one set of the vertices should be drawn at fixed positions (anchors) while the vertices of the other set should be drawn freely at suitable positions near the anchors (see Figure 3.4). This results in a circular shaped graph. A similar graph can be constructed with the method of Dumas et al. [DMRW11], where one set of vertices is placed in the middle of a circle and the other set around them using an edge bundling technique to reduce the number of edges. These methods assume that one set of vertices is rather small and thus cannot be used for our purpose.

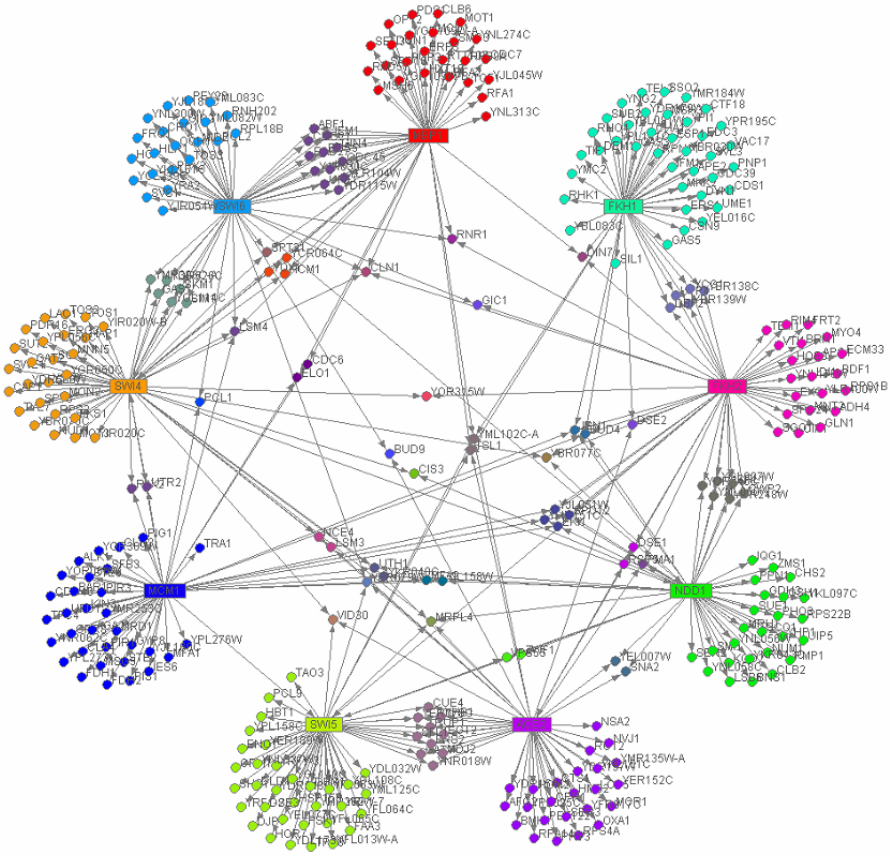
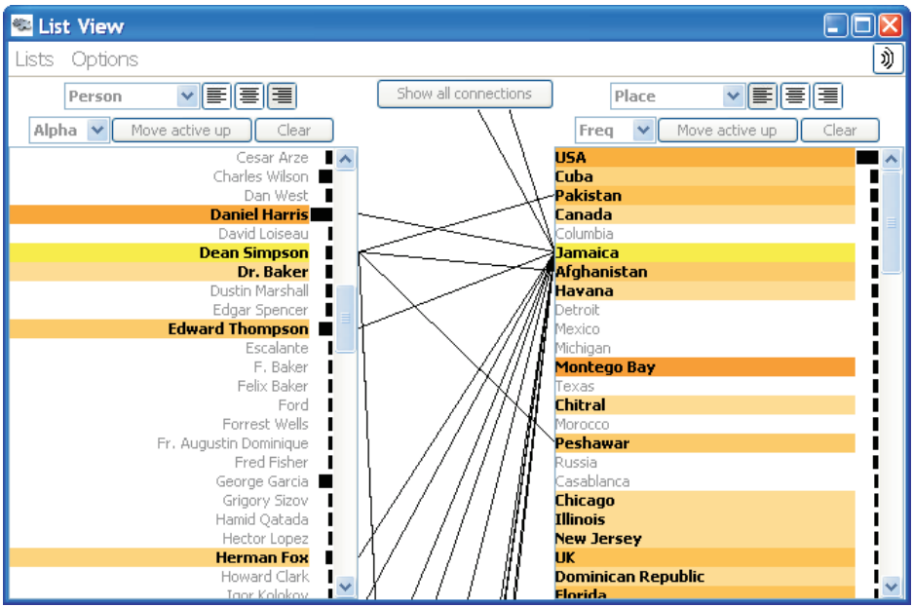


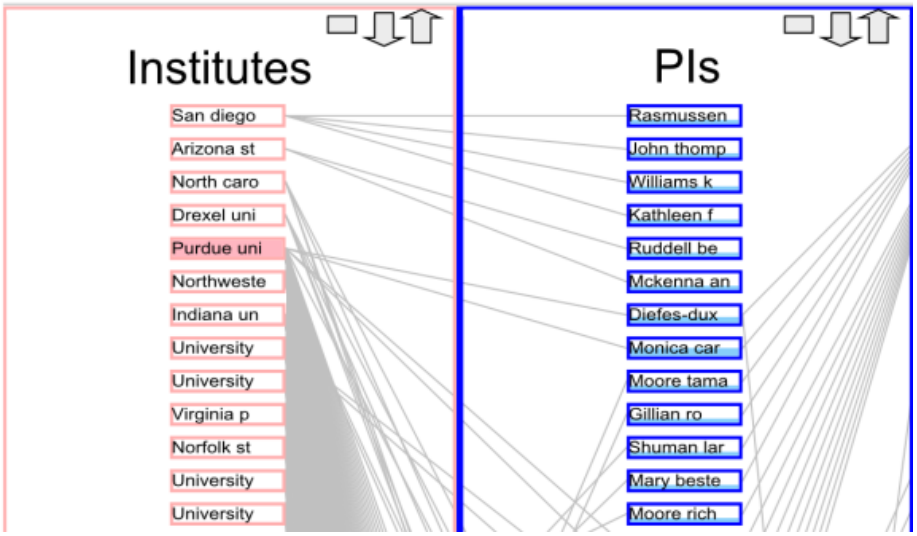
Figure 3.4: Example of bipartite graph visualization with the method proposed by Misue et al. [Mis06].

Another typical visualization type is the use of two lists representing the two sets of nodes and then showing the connection between the nodes through edges. Such techniques were proposed by Stasko et al. [SGL07] as part of their visualization tool *Jigsaw* (see Figure 3.5a), Ghani et al. [GKL+13] in their *Parallel node-link bands* (see Figure 3.5b), or by Schulz et al. [SJUS08] in their visual analysis framework for biological networks. *Jigsaw* is a visual analytic toolkit for exploring documents and their relationships and consists of four connected views. These views are a node-link diagram, a scatter plot, a text view, and the already mentioned list view. In Figure 3.5a, one list shows people and the other one places. Multiple entities can be selected within each list and their corresponding relations are highlighted in orange, where the strength of the connection is encoded in the color's brightness. Additionally, lines are drawn between connected entities. If many entities need to be displayed, the list becomes scrollable.

PivotPaths by Dörk et al. [DRRD12] does not only use two lists, but also adds a third between them representing the attribute shared by the other two. This results in a user interface that lets one explore the data as a tripartite graph. The interface is



(a)



(b)

Figure 3.5: Two example of list views: (a) part of *Jigsaw* by Stasko et al. [SGL07], (b) *Parallel node-link bands* by Ghani et al. [GKL+13].

divided into three regions, each representing a different set of the graph. Figure 3.6 shows how these regions are displayed in *PivotPaths*. The three regions thereby represent people, resources, and concepts. The middle region shows samples of resources, which act as main exploration objects. The other regions give additional information about the resources and the size of their elements scale according to the number of connections to

the resources. This concept can also be changed and people or concepts can be switched with the middle region to explore those samples, respectively. The resources of interest can be filtered by using any entity of the three regions as an anchor. If, for instance, one person of the people-region is set as anchor, only connected resources and their concepts will be displayed. If there are too many samples of resources to display, the number will be automatically limited by showing only the most recent resources, the most cited resources, or a random subset of the resources.

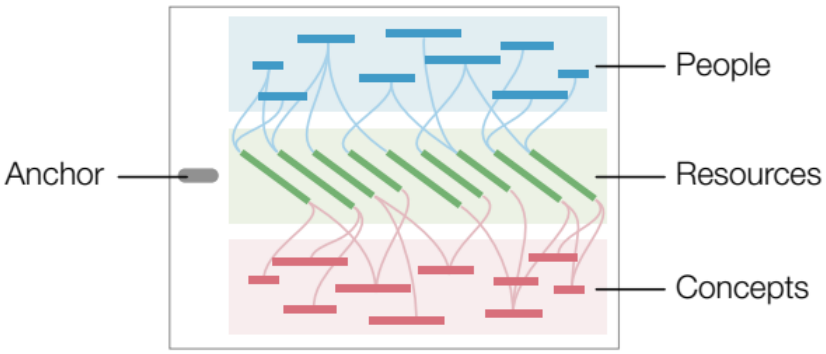


Figure 3.6: Concept of *PivotPaths* by Dörk et al. [DRRD12].

The limitation of these list views lies in their scalability if a huge number of entities need to be displayed. The problem can be solved either by offering the possibility to scroll large lists like in *Jigsaw* or by only displaying a subset of the data like in *PivotPaths*. The visual analysis framework by Schulz et al. [SJUS08] offers additional features to overcome the problems of visualizing large sets of entities and edges. For very long lists, they use a focus+context approach where all list elements that are not hovered will be minimized, resulting in a reduction of the overall list height. For large sets of edges, they use edge crossing minimization algorithms and the possibility to highlight edges.

3.3 Aggregation

If a graph with several thousands of nodes and edges needs to be displayed, showing it as a whole is no option. Exploration will be nearly impossible because of edge clutter and performance reasons. The graph therefore has to be abstracted to reduce its size and offer a overview, while still maintaining the possibility to explore particular elements of the original graph. This also follows Ben Shneiderman's [Shn03] visual information seeking mantra "overview first, zoom and filter, then details on demand". This abstraction can happen in data space or visual space [EF10]. While abstraction in data space uses the original data objects to generate new abstracted items, abstraction in visual space only abstracts the visual representation of visible data objects.

Von Landesberger et al. [vLKS+11] defined *filtering* and *aggregation* as two approaches to abstract and consequently reduce the size of graphs. The concept of filtering or sampling

is that it reduces the size by removing nodes and edges of the graph. This can happen either as a preprocessing step, where certain data elements are removed before the graph is visualized or during the visualization process, where only certain node and edges are displayed. The criteria which nodes and edges to remove can be based on different attributes. For instance, a minimum edge weight or minimum centrality can be used to determine which edge or node to remove. There are also stochastic filtering methods that randomly remove nodes and edges while still maintaining the overall properties of the original graph [LF06].

With aggregation on the other hand, nodes and edges are grouped together to new meta nodes and edges. By iteratively aggregating the newly created groups of nodes, a hierarchical tree-like structure of aggregated elements can be created. Elmquist and Fekete [EF10] refer to this structure as *aggregation tree*. The hierarchical aggregation can be accomplished by using a bottom-up or top-down approach. While bottom-up techniques start with a single element and consecutively aggregate similar elements together, top-down approaches start with the aggregation of all elements and iteratively split them up.

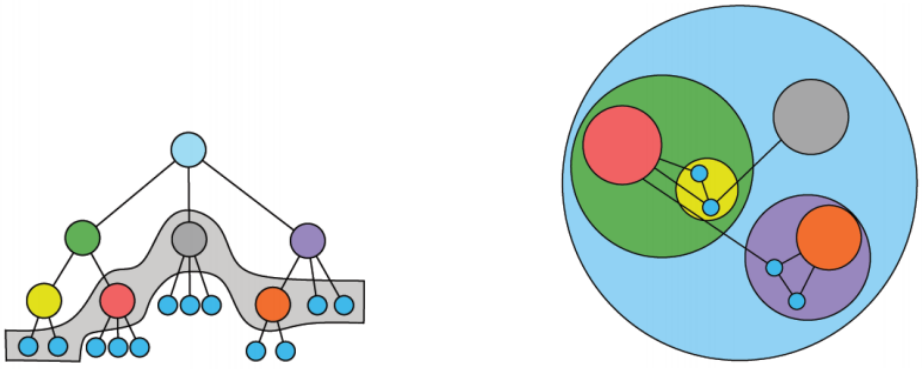
To display these aggregation trees, a suitable visualization technique must be provided. This technique must not only be able to handle hierarchical structures, but also offer a way to represent the underlying aggregated elements. To visualize these underlying elements in an abstracted way, certain derived characteristics can be used. Andrienko and Andrienko [AA06] have defined some of the most frequently derived characteristics of these aggregated elements:

- the number of aggregated elements,
- the accumulated sum of the aggregated elements' data values,
- the mean of the aggregated elements' data values,
- the value range of the aggregated elements,
- the most frequent value occurring within the aggregated elements, and
- the median of the aggregated elements' data values.

Depending on the underlying data structure, different visualizations types can be used to offer the possibility of visualizing and exploring aggregation trees. Elmquist and Fekete [EF10] defined *overlapping* and *space-filling* visualizations as the two main layout types. While overlapping visualizations are not restricted by a predefined layout, thus resulting in possible overlaps of elements, space-filling techniques are. Typical types of the former are scatterplots, node-link diagrams, or parallel coordinates, while adjacency matrices or treemaps are examples of space-filling approaches. Since we are dealing with graphs, we will focus on visualization techniques used in node-link diagrams and matrix representations. However, in Section 3.5 we will also show examples of concepts used

in *Parallel Coordinates* due to the fact that they can be adapted for list approaches presented in Section 3.2.

An example for exploring hierarchical graph structures using node-link diagrams is *GrouseFlocks* proposed by Archambault et al. [AMA08]. *GrouseFlocks* enables users to interactively construct graph hierarchies by searching and selecting attributes of the underlying graph data. With those selections, an aggregation tree based on similarities of the selected attributes is constructed. *GrouseFlocks* then visualizes certain cuts in the hierarchy of the aggregation tree as superimposition on top of the graph (see Figure 3.7). The user interface consists of a tree-view in form of a list and a graph-view similar to the one in Figure 3.7b. Users can navigate inside the aggregation tree by either interacting with the tree- or the graph-view. Through their interaction, different cuts, i.e., thus different levels of the aggregation tree are visualized. The main disadvantage of *GrouseFlocks* is that it heavily relies on data attributes. If a graph only consists of nodes and edges and lacks additional node attributes, it will hardly be possible to explore it with *GrouseFlocks*.



(a) (b)

Figure 3.7: Example of a graph hierarchy cut: (a) original graph with cut (gray curve) and (b) graph cut superimposed on top of the graph [AMA08].

An approach that is similar to *NodeTrix* described in Section 3.1 is *TreeMatrix*. *TreeMatrix* is a matrix-based approach proposed by Rufiange et al. [RMF12] for exploring hierarchical graph structures. It combines several different visualization techniques. For visualizing subtrees of the aggregation tree, it uses collapsible matrices. Figure 3.8 shows the user interface of *TreeMatrix* with a detail view of an opened adjacency matrix. The black squares inside the matrix indicate its tree structure. The lists on the left and top represent the nodes of this subtree in gray, whereas further subtrees and their children are colored in yellow. Edges between nodes are depicted by their corresponding matrix cell, where the color encodes their edge weight. Additionally, they are represented by arcs between nodes at the list on top. These can also lead to other opened or closed matrices. Although using collapsible matrices representing subtrees is a space-saving concept, it faces problems if a

subtree with many nodes needs to be opened. Furthermore, there can be also problems with visual clutter if many edges lead from one matrix to another one.

The following approach could be used to overcome the problem of effectively visualizing a large number of edges. In case of aggregating edges, the concept of edge bundling was introduced. Edge bundling approaches exist for hierarchical graph structures [PXYH05] as well as for general graphs. The concept is either based on using the hierarchal properties or a control mesh [CZWL08] to bundle the edges. However, Holten and van Wjik [HvW09] proposed a self-organizing edge bundling approach, where edges are modeled as springs. An attracting force is used between these springs to enable them to form a bundled structure (see Figure 3.9). This approach is neither dependent on a hierarchical graph structure nor on a generated control mesh.

3.4 Biclustering in Visualization

In case of bipartite graphs, biclustering (see Section 2.3) can be used as aggregation method. Parallel to the development of biclustering algorithms, new visualization approaches were proposed that made use of theses biclustering techniques. One often used method is a simple colored matrix-based visualization of the underlying biadjacency data matrix. Barkow et al. [BBP$^+$06] proposed *BicAT*, a tool for analyzing bipartite data (see Figure 3.10a). It has different clustering algorithms implemented and offers a matrix visualization were each element is colored according to its data value. This makes it easier to spot and select found clusters. Similar approaches have been suggested by Filippova et al. [FGK12], who developed *Corel*, a suite for comparing clusterings, and Sun et al. [SNR14], who suggested a matrix-based visualization in their five-level design framework (see Figure 3.10b).
The limitation of this kind of visualization lies in its scalability, since it becomes very difficult to illustrate and explore larger matrices with thousands of rows and columns.

Another visualization technique that makes use of biclustering was proposed by Sun et al. [SMNR16] and Onoue et al. [OKSK16], who use edge bundling approaches to minimize clutter and edge overlap (see Figure 3.11). Due to the fact that only edges are bundled and not nodes, they face the same problem if several thousand nodes need to be visualized.

Santamaría et al. [STQ08] proposed *BicOverlapper*, a tool designed to visualize biclustered gene expression data. The idea is to display the data as a graph where nodes represent either genes or conditions, which are interconnected through edges if they share the same bicluster (see Figure 3.12a). Biclustered nodes are then enclosed in a semi-transparent hull and the edges between the nodes are removed to avoid visual clutter (see Figure 3.12b). Since these hulls are basically *Euler diagrams*, they face the same problem for finding a suitable visualization if a group of nodes share more than three hulls [RD10].

The concept of *NodeTrix*, mentioned in Section 3.1, was also picked up in more recent publications that are dealing with the visualization of biclustered data such as *BixPlorer*

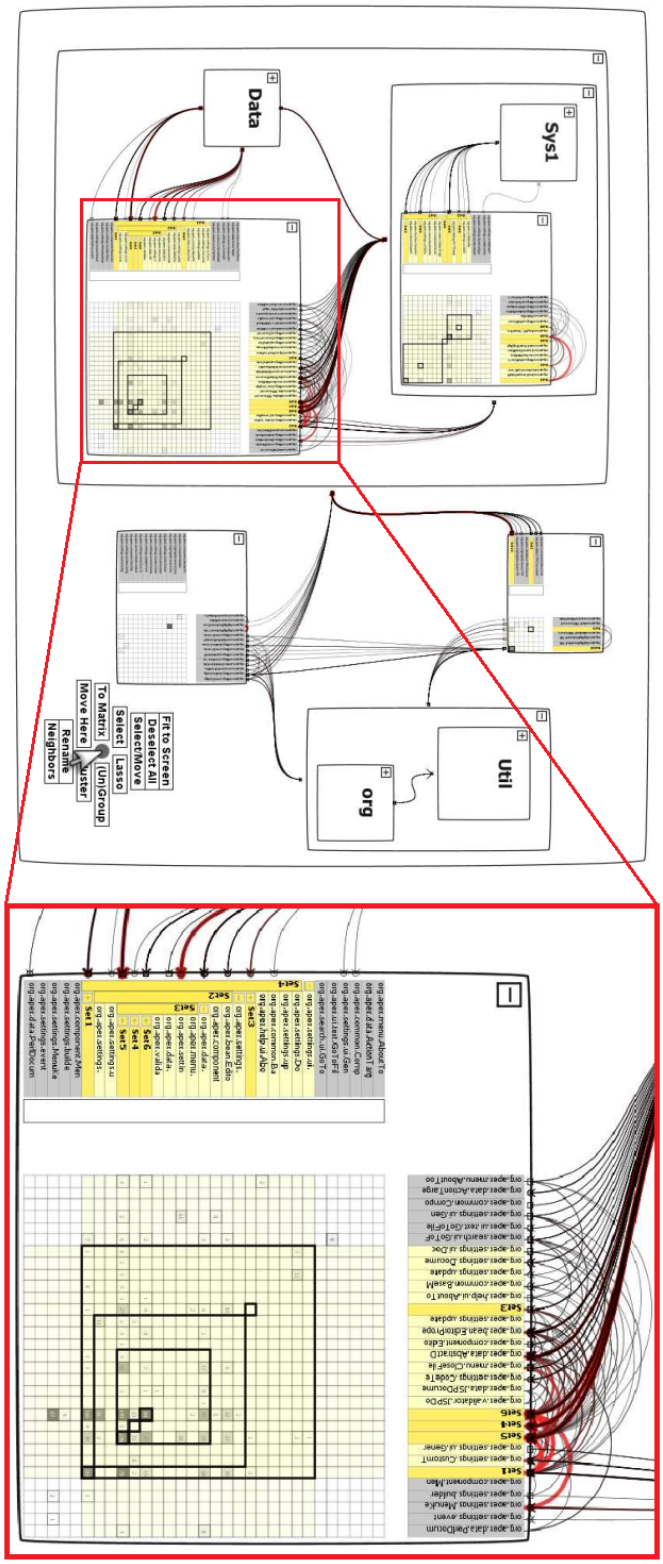
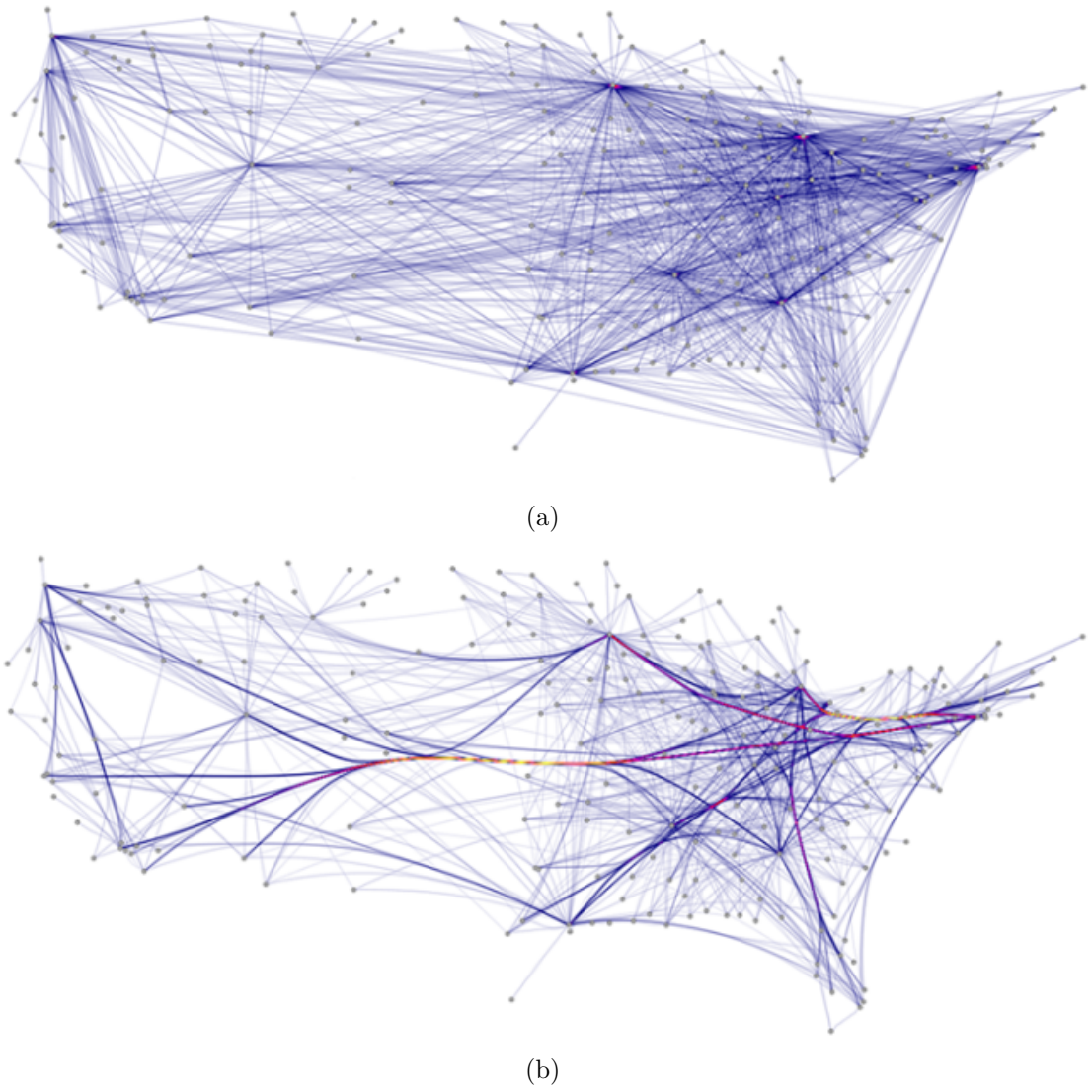


Figure 3.8: Prototype of *TreeMatrix* proposed by Rufiange et al. [RMF12] with a detail view of an opened adjacency matrix.

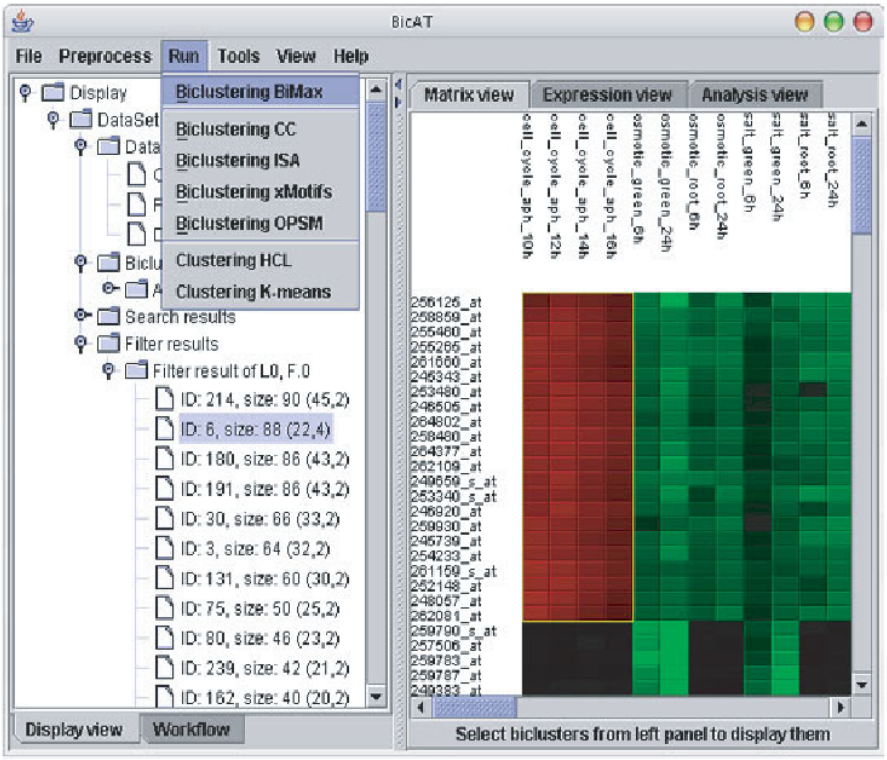


(a)

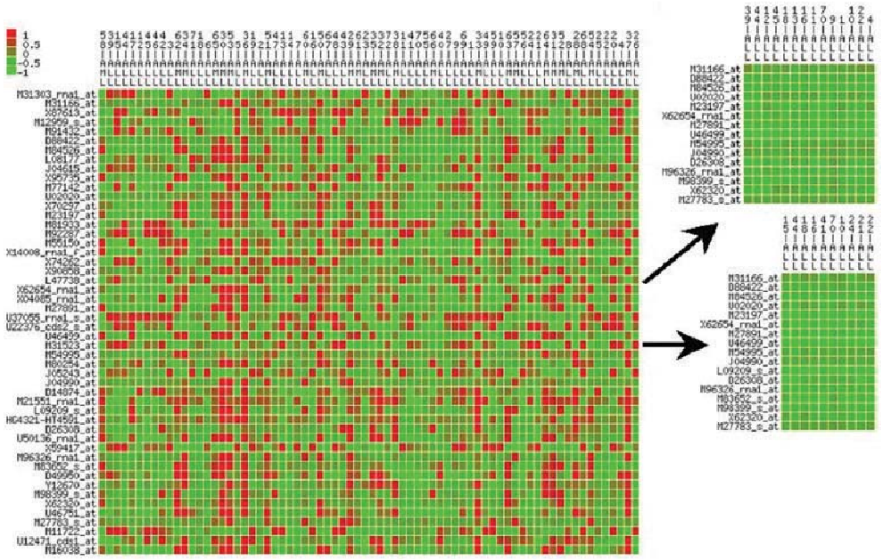
(b)

Figure 3.9: Example of force-directed edge bundling proposed by Holten and van Wjik [HvW09]: (a) original graph, without bundling and (b) graph with bundled edges.

by Fiaux et al. [FSB+13], *Furby* by Streit et al. [SGG+14] or the exploration interface developed by Xu et al. [XCQS16]. All of them use the *NodeTrix* concept in a similar way by representing the biclustered sub-matrices as nodes in a node-link diagram. *BixPlorer* relies completely on this concept and was developed for finding patterns in textual datasets on a high resolution display (10240×3200 pixels). *Furby* additionally uses a combination with heatmaps and bar charts to offer the possibility to further examine single clusters. Whereas Xu et al. [XCQS16] use treemaps to reveal attribute patterns within clusters, besides utilizing the concept of *NodeTrix* for relational patterns between



(a)



(b)

Figure 3.10: Examples of matrix-based visualizations: (a) *BicAT: A Biclustering Analysis Toolbox* proposed by Barkow et al. [BBP+06], (b) data matrix with two mined biclusters suggested by Sun et al. [SNR14].

Figure 3.11: Example of edge bundling in *BiSet* proposed by Sun et al.: (a) original edges, (b) after semantic edge bundling [SMNR16].

clusters. A limitation of these approaches are the number of biclusters and also their size. If there are many biclusters with numerous rows and columns to visualize, this will result in large adjacency matrices taking up a lot of screen space.

BiDots, a very recent approach by Zhao et al. [ZSCC18] tries to overcome this limitation by displaying biclusters in separate rows (see Figure 3.13). A row thereby consists of two sets of entities and a rectangle where their relationship is encoded. Entities are represented as gray circles with a unique line pattern. The darker the circle, the more often it is shared by different biclusters. Between the two sets of entities, vertical orange line strips in a rectangle represent the weighted relationships between those entities. The position of the line strips reflects the weight of each relationship. This means that lines on the left side of the rectangle represent lower and on the right side higher weights. To explore the data, *BiDots* offers different interaction techniques. Besides reordering rows and fading out unutilized entities or relationships, it is also possible to pin certain entities. This results in a new column showing only the pinned entity colored in blue. All other entities connected to the pinned one are also colored blue to emphasize their connection. By representing biclusters as rows, it scales better if visualizing datasets

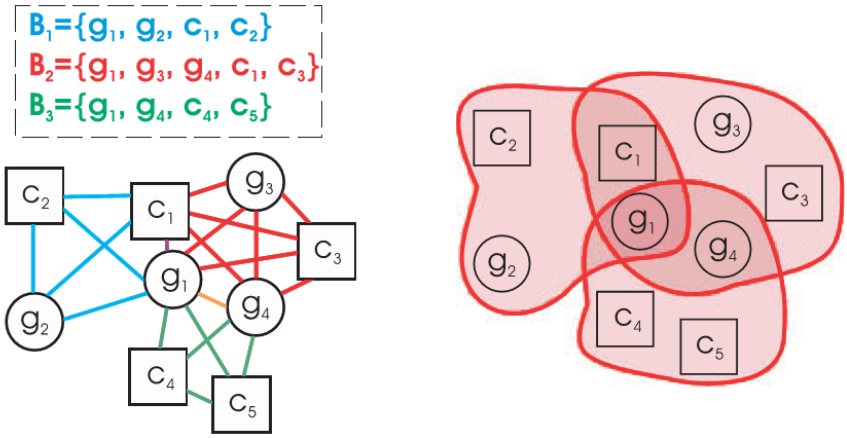


Figure 3.12: Concept of *BicOverlapper* proposed by Santamaría et al.: (a) sets of biclustered vertices and their corresponding interconnected graph, (b) semi-transparent hull enclosing the biclustered vertices [STQ08].

with many biclusters. However, it has limitations if there are a lot of unique entities within a bicluster, due to the fact that all containing entities are visualized side by side in the corresponding row.

3.5 Sankey Diagrams and Parallel Sets

The following two approaches are not explicitly designed for bipartite graphs. However, they both use a similar visual encoding for quantitative data that we adapted for our implementation of BiCFlows.

Sankey Diagrams can be found in very early illustrations, like "Napoleon's Russian campaign of 1812" by Charles Joseph Minard (see Figure 3.14). Here, he visualized not only the geographic position of Napoleon's troops at certain moments in time, but he also encoded the size of his army. Minard did this by mapping the number of troops to the thickness of the lines pointing from one geographic location to another one. This type of diagram, where flows and their quantitative transformations are visualized, is called *Sankey Diagrams*. The underlying data structure of *Sankey Diagrams* are directed weighted graphs, where the edge weight is represented by the thickness of the line in the diagram. The graph thereby follows a specific flow property, where the incoming sum of edge weights for each node equals the outgoing sum of edge weights [RHF05].

As described in Section 2.2, edges of a bipartite graph only connect nodes from one set with nodes of the other one. Thus, these edges can also be drawn as directed edges and furthermore be interpreted as a flow. Because the nodes of these two sets have only outgoing or incoming edges and not both, the flow property described above also fits

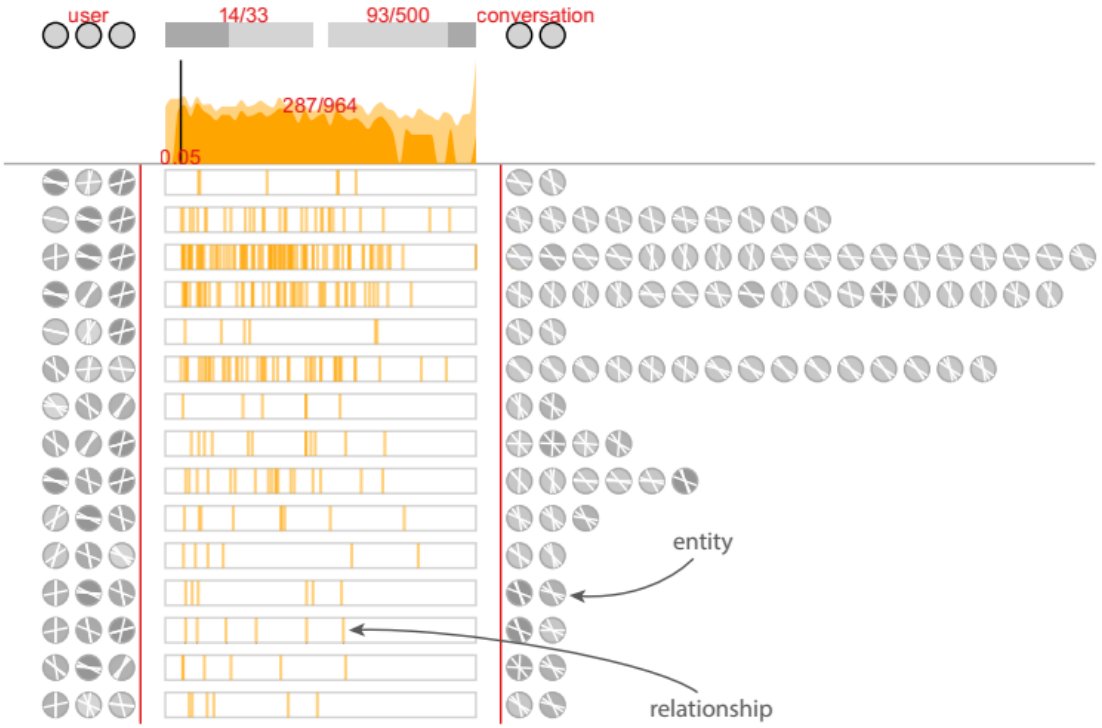


Figure 3.13: User interface of *BiDots*. Rows represent biclusters, circles represent unique entities and orange line strips represent weighted relationships between entities [ZSCC18].

for bipartite graphs. For this reason, the concept of *Sankey Diagrams*, where the edge weight is mapped to the thickness of the edges and the summed edge weights of a node to it size, can also be applied to bipartite graphs.

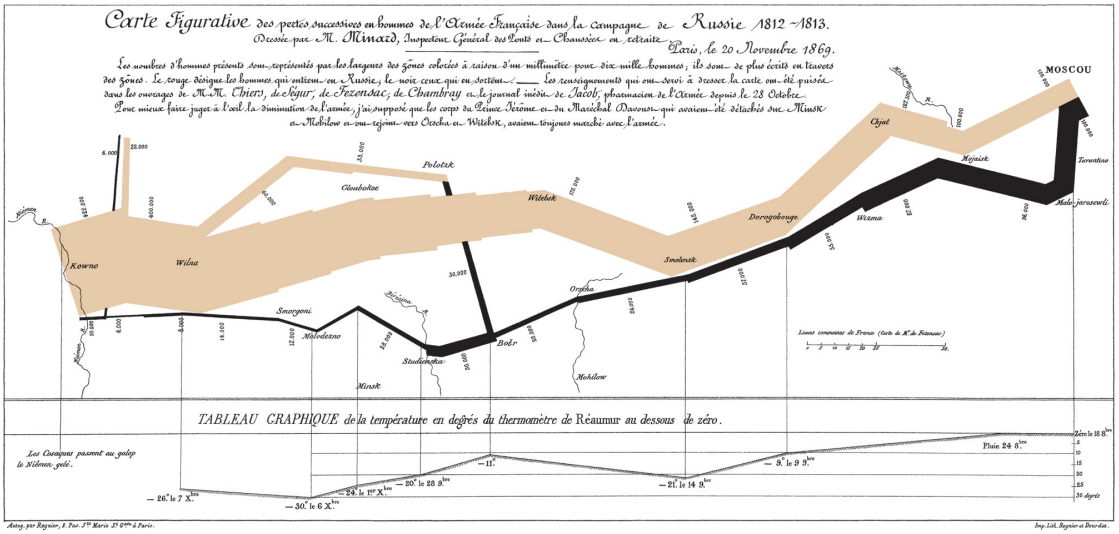
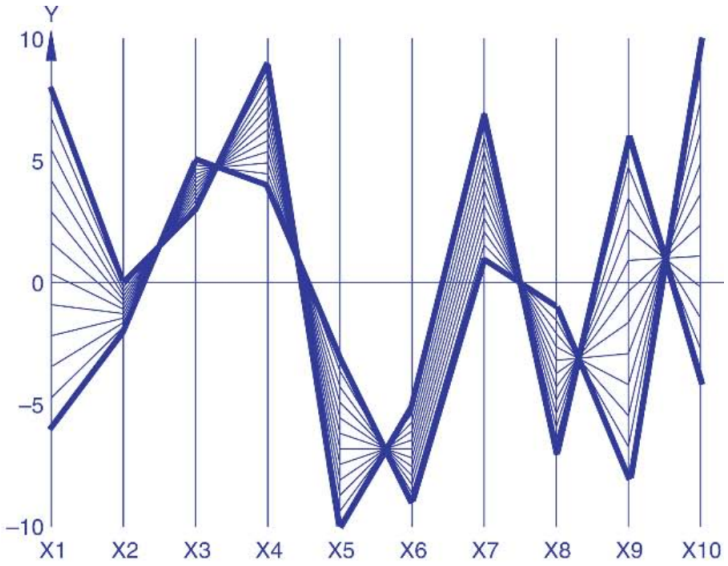


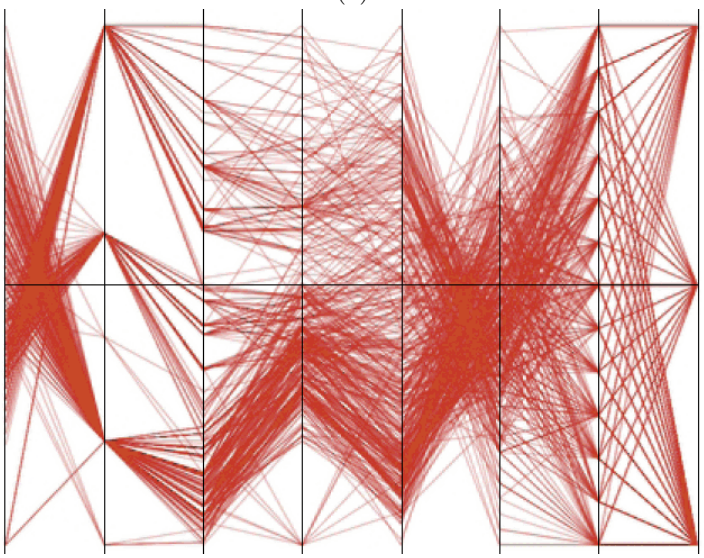
Figure 3.14: "Napoleon's Russian campaign of 1812" drawn by Charles Joseph Minard in 1869 [KM13].

Parallel Coordinates is a technique to visualize multidimensional data. Every dimension is thereby represented as a vertical axis. A data object is then visualized as a polygonal line connecting every axis at a certain value. Figure 3.15a shows an example of *Parallel Coordinates* where ten dimensions are visualized. A problem arises if many data objects need to be visualized, which often results in clutter (see Figure 3.15b). Another drawback of *Parallel Coordinates* is, that it is not suitable for categorical data.

To overcome these limitations, *Parallel Sets*, an extension of the *Parallel Coordinates* layout was proposed. Bendix et al. [BKH05] describe their *Parallel Sets* approach as a combination of the advantages of *Parallel Coordinates* where dimensions are treated independently and includes the possibility to use frequency-based categorical data. The idea of *Parallel Sets* is that instead of continuous axes, sets of boxes are used. Each box represents either a category or aggregated data points on this specific axis. The size of each box scales with the frequency of the category or the number of items aggregated, respectively. The thickness of the polygonal bands connecting each of these boxes also scale in the exact same manner. Figure 3.16 shows an example of *Parallel Sets*, where a dataset of Titanic passengers is visualized. It shows the three dimensions *Class*, *Sex*, and *Survived*, where each dimension has different categories. In this example the categories of *Class* are colored all differently to follow their connections more easily. For example, one can see that all crew members are male and the majority of them did not survive. Bendix et al. [BKH05] also offered the possibility to interactively analyze the data. These interactions include adding, removing, merging, or splitting categories, reordering dimensions and categories as well as highlighting connected polygonal bands.



(a)



(b)

Figure 3.15: Examples of *Parallel Coordinates* with (a) ten dimensions [Ins09] and (b) eight dimensions (cluttered) [EF10].

3.6 Comparative Evaluation

Since one of our contributions is also a comparative evaluation between a clustered and an unclustered approach, we show some related work here.

Hearst and Pedersen [HP96] evaluated the document browsing technique *Scatter/Gather*.

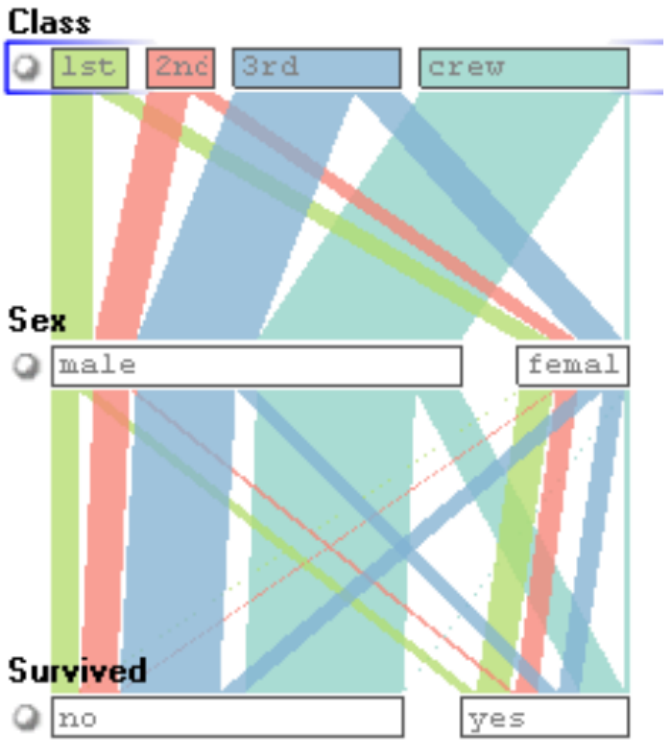


Figure 3.16: Example of *Parallel Sets* by Bendix et al. [BKH05].

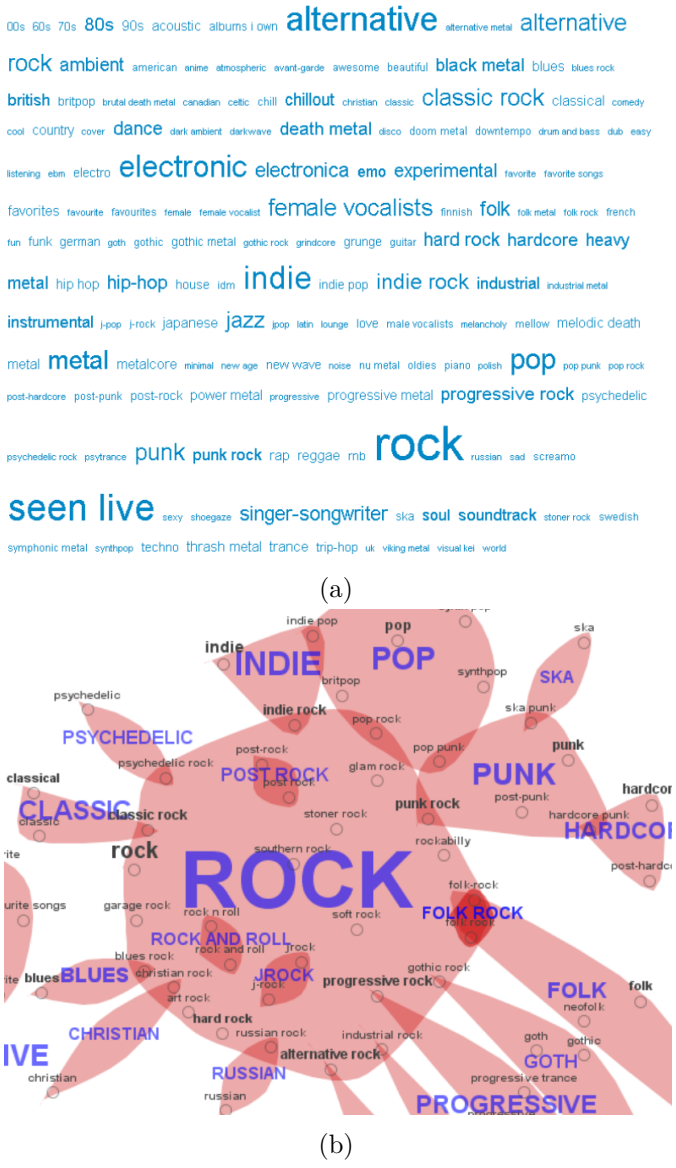
Scatter/Gather clusters documents based on their topical similarity. For each cluster, a textual summary is provided. Theses summaries consist of terms derived from the documents' topics inside these clusters. The concept of *Scatter/Gather* can be used in combination with a preceding query, where the search results are clustered afterwards. Since these clusters are topic-coherent, they can support users to find documents on a certain topic more easily. Furthermore, it would be possible to cluster the documents of a cluster again to split them into new topically similar groups.
In their evaluation, they first investigated if this clustering approach really does group together more relevant documents. They took the top n documents of a query result and used *Scatter/Gather* to cluster them into five clusters. They used the containing number of documents per cluster as ranking measure, where the cluster with the most documents was ranked best. Then they ordered the documents from the highest rated cluster by closeness to the query and by closeness to the cluster centroid and compared them to an equivalent number of documents from the top n documents. For $n = 100, 250, 500$ and 1000, their results showed that the clustering does indeed reveal more relevant documents than an equivalent number of documents from the top of the originally unclustered query result.
In a user study, they further examined how often users chose the cluster with the most documents based on its summary. The task was to find as many relevant documents

for certain topics as they can in 30 minutes. In their results and discussion, Hearst and Pedersen showed that on the one hand users were able to interpret the summaries, and on the other hand, to find the cluster with the most relevant documents. Moreover, users mentioned that they found the concept of clustering useful to determine groups of similar topics.

Chen et al. [CSBT09] proposed *TagClusters*, a semantic clustering concept for tags. Tags are a way to categorize and organize digital items and are provided by many online communities, where music or photos are shared, for instance. *TagClouds* are a way to visualize popular tags with different font-sizes or colors depending on their frequency or newness. However, they lack hierarchical relations and can have linguistic issues such as different meanings for the same tag. *TagClusters* tries to overcome these problems by clustering tags based on their semantic similarity.
To evaluate their approach, they conducted a user study, where they compared *TagClusters* to *TagClouds*. They used a repeated measures within-subjects design with 12 participants and a task-based evaluation approach. This means that every user had to perform six predefined tasks with each interface, where the completion time, answer precision, task easiness, and usefulness of the current system were recorded. Their evaluation results revealed that grouping together semantically similar tags helped users to discover smaller tags, which were not that visually prominent in *TagClouds*. As an example, they mention the sub-genres of "rock", which were hardly noticed due to their small font-size in *TagClouds* (see Figure 3.17a), but are better visible in *TagClusters* (see Figure 3.17b), because of their closeness and connection to the "rock"-cluster. Users were also able to determine relationships between different genres better with *TagClusters* than with *TagClouds*.

Cao et al. [CGSQ11] conducted a user study where they investigated if their developed approach *DICON* supports users in comparing and interpreting clusters. *DICON* is a multidimensional cluster visualization where statistical information is embedded in icons. They use a treemap or Voronoi-like layout within the icons to represent the different dimensions. Each dimension is colored differently and the number of data objects clustered is represented by the size of the icon. In their design guidelines, they suggest to rearrange the positions of each dimension so they are equal for every icon and thus more comparable.
In a case study, they compared *DICON* to other multidimensional visualization approaches like scatterplot-matrix and parallel coordinates, by using a dataset of 407 cars with seven dimensions. Figure 3.18 shows the results of visualizing this dataset with parallel coordinates and *DICON*, respectively. *DICON* not only encodes the data in a compact form, but also immediately reveals the size of each cluster, which is not possible with parallel coordinates. Through the consistent arrangement of dimension within the clusters, it is also possible to see that European and Japanese cars share very similar features compared to American cars.

Cao et al. also conducted a user study, where they compared three types of cluster-icons with each other. One layout used random order packing (see Figure 3.19a), one used



(b)

Figure 3.17: Example of (a) TagCloud and (b) *TagClusters* by Chen et al. [CSBT09].

ordered packing following the *DICON* design guidelines (see Figure 3.19a), and the last one additionally encodes the statistical distribution in the shape of the cluster-icons (see Figure 3.19c). They used a between-subjects design for their user study, where they split the 30 participants into three groups of 10 people. Each group was presented with a different type of cluster-icon and had to perform two predefined tasks for two different datasets. One dataset consisted of 300 entities within nine clusters and the other one of 1000 entities within 50 clusters. Their tasks were to find similar cluster in one dataset and groups of similar clusters in the other. They recorded the task completion time and

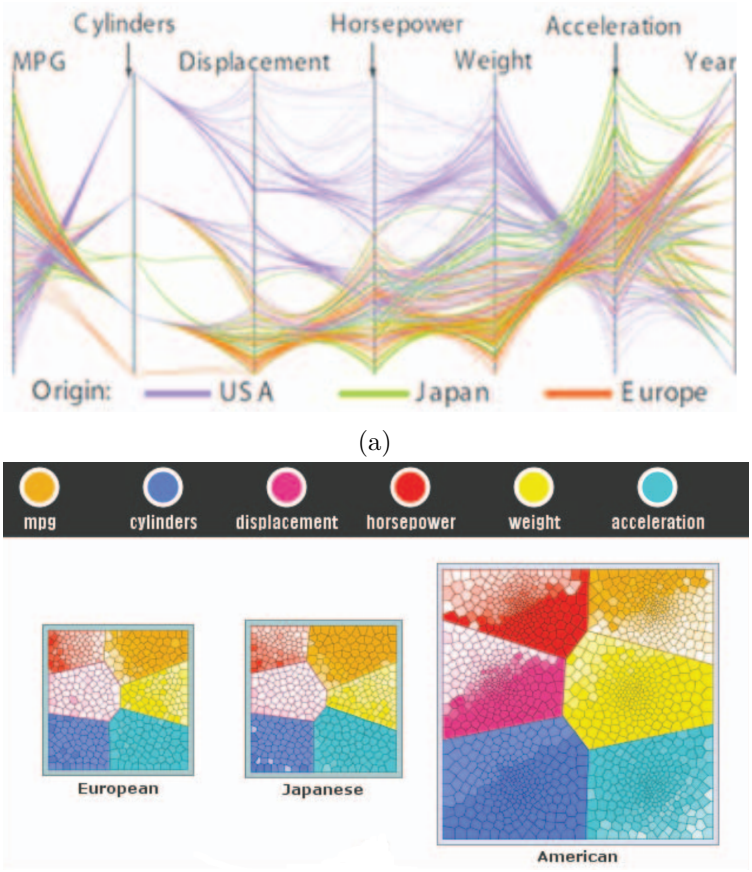


Figure 3.18: Dataset of 407 cars with seven dimensions visualized with (a) parallel coordinates and (b) *DICON* [CSBT09].

the success rate during the study and conducted a usability questionnaire afterwards. Their results showed that their proposed packing types outperformed the random packing type in both, completion time and success rate. Users were able to successfully compare and identify similar clusters even with the larger dataset.

In all conducted user studies, where clustered approaches were compared with unclustered ones, clustering could reveal more information when looking for similar data entities. It was described as a good approach for initially grouping together entities to get a better overview of the data. Especially for large datasets, where it is not possible to display everything at once, clustering showed to be a good choice for aggregation and size reduction.

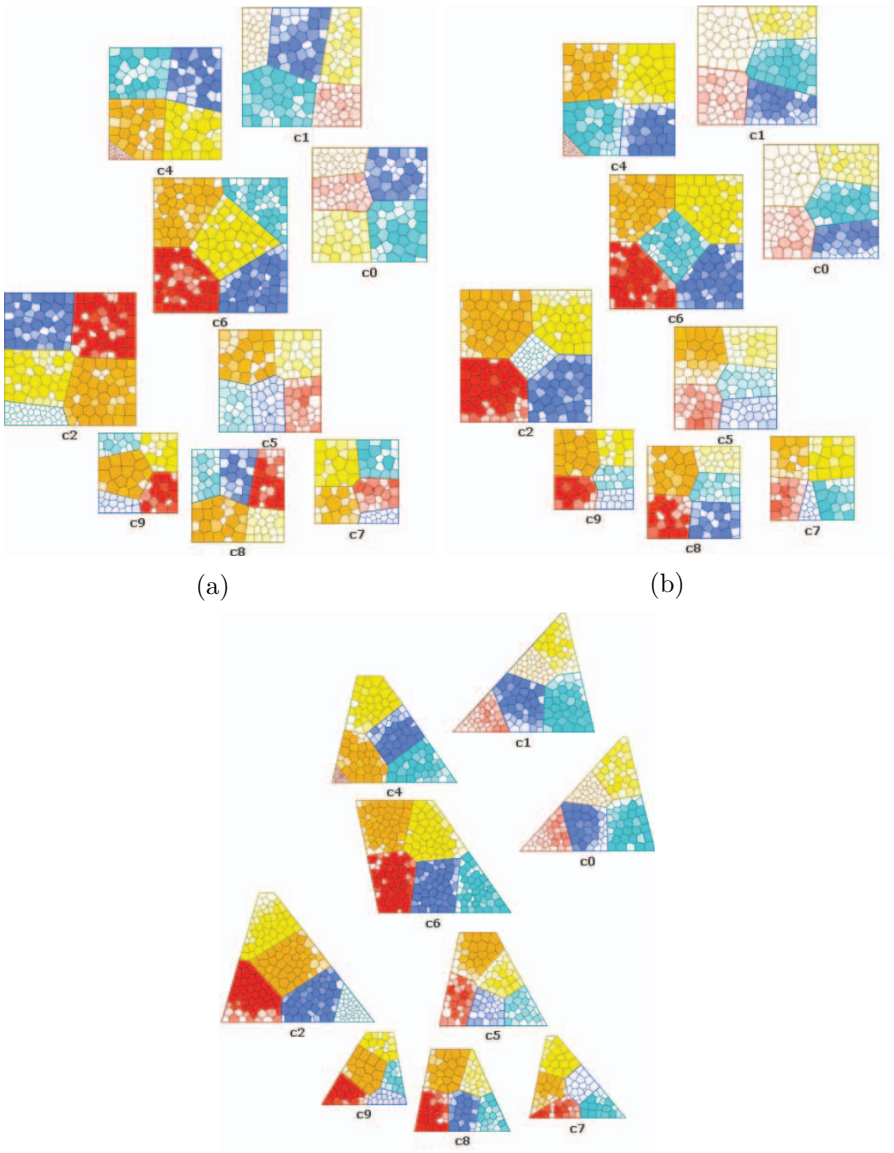


Figure 3.19: Three examples of cluster-icons used for the user study: (a) random order packing, (b) ordered packing using the *DICON* design guidelines, and (c) ordered packing with different shapes [CGSQ11].

CHAPTER 4

Visualization and Interaction

This chapter describes *BiCFlows*, a novel visual exploration interface for large bipartite graphs. The main idea is to overcome the limitations of existing visualization approaches when dealing with bipartite graphs that contain thousands of nodes and edges. As discussed in Chapter 3, these approaches do not scale effectively if large datasets are used. Moreover, the goal is to support non-targeted exploration, where the user discovers information without a predefined question in mind. It should allow users to find negligible entities without explicitly looking for them and follow Thomas and Cook's mantra *"detect the expected and discover the unexpected"* [CT05, CT06].

Because of its properties as a bipartite graph, it makes sense to split the nodes into two sets, which is a useful concept also applied in existing approaches (see Section 3.2). Moreover, since the underlying data represents directed edges from nodes of one set to nodes in the other set, it lends itself to a parallel alignment from left to right, which emphasizes the reading order. These two parallelly aligned sets can be displayed as two lists of nodes. This results in an approach similar to *BiSets* and *JigSaw* mentioned in Chapter 3. Since the edges between nodes of these two sets are also weighted and edge weight is often encoded in terms of line width, it makes sense to adopt this concept for the parallel lists as well. This means that edges with a higher edge weight will be drawn thicker than the ones with a smaller weight. Since the nodes in each set are represented as list elements, the height of these elements can also be used to encode weight. The weight of a node is determined by the sum of its edge weights. This sum can then be used to define the height of each list element. In case of the Media Transparency Database, we are dealing with cash flows. So, each edge weight is represented by the amount of money going from list elements of one set to list elements in the other set. The smallest edge weight is € 5000, and if all edge weights from one set to the other one are added together it results in the total money exchanged, which is € 1100 million. We can use this information to think about the appropriate display size needed to visualize all these list elements. If the smallest weight is represented by only 1px in height, we would still

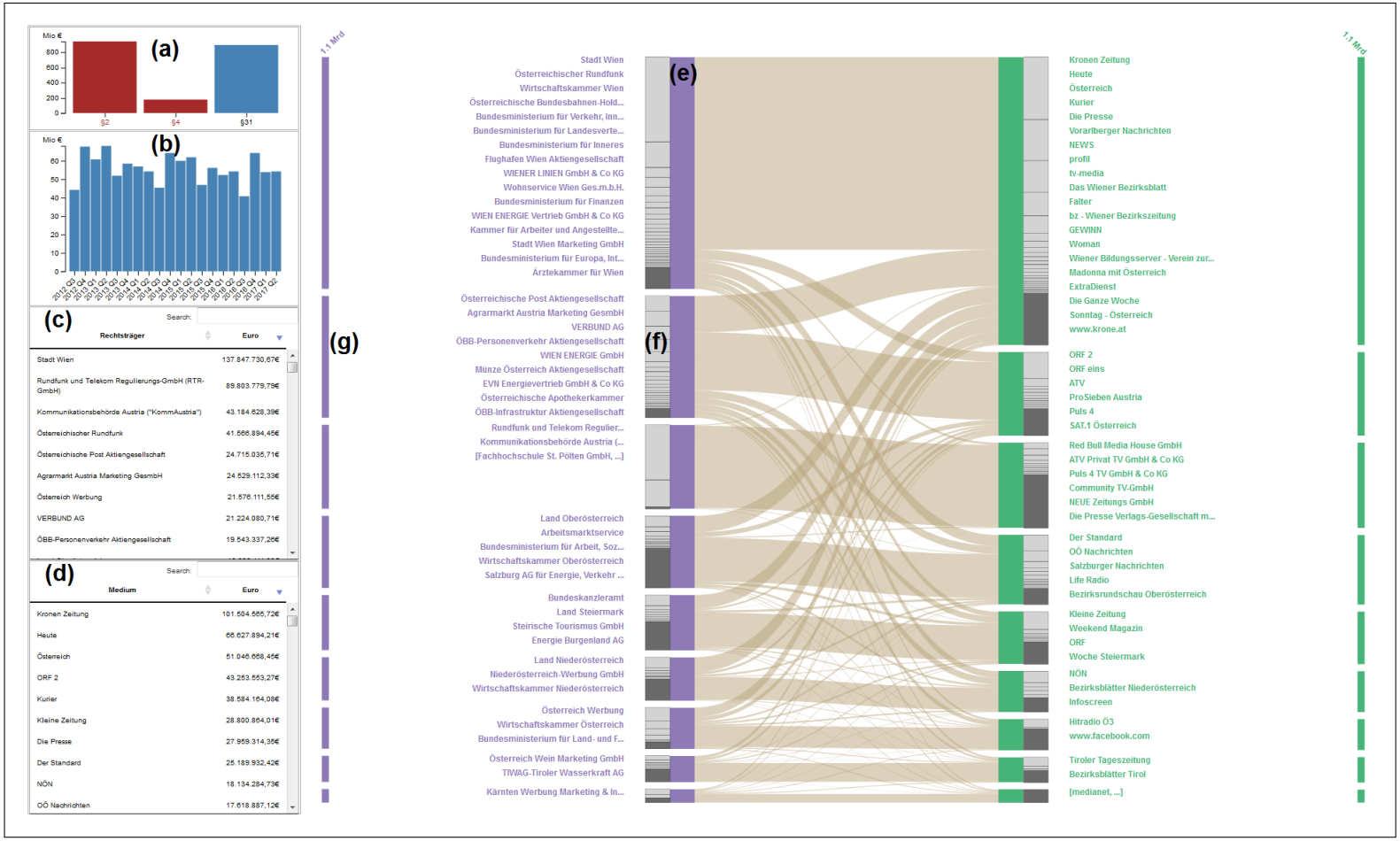


Figure 4.1: BiCFlows: (a) bar chart of legal base, (b) bar chart of quarters, (c) table of legal entities, (d) table of media organizations, (e) cluster bar, (f) stacked bar, (g) context bar.

need 220,000px to display every element. This fact makes it neither possible to visualize nor to effectively explore the data.

This is where aggregation, a commonly used approach in big data visualization [AA06], comes into play. If there are already inherent hierarchies present in the dataset, they can be used to aggregate the data and reduce the number of initially visible elements. If like in the Media Transparency Database, these hierarchies are missing, other aggregation concepts have to be used. One of the most often applied approaches is clustering [EF10]. Consecutive clustering can be used to create an artificial hierarchy within the data, based on existing data object properties or their relationships. Since we are dealing with bipartite graphs, the concept of biclustering can be applied. With the help of biclustering, list elements are hierarchically aggregated to clusters, where each cluster will be split into smaller subclusters upon user interaction. The idea is to aggregate all elements to a limited number of clusters. These clusters are visualized initially to provide an overview between cluster interconnections. Every cluster can then be further explored through user interaction. Here, the exploration is accomplished by clustering existing clusters again. This results in a cluster hierarchy where single elements form the lowest level.

As biclustering algorithm, we used *CoClust* proposed by Ailem et al. [ARN15] (see Section 2.3), but any other algorithm that can handle weighted biadjacency matrices and assumes an underlying block diagonal structure (see Figure 2.4) can be used. The structure is important, since every list element should only be part of exactly one single cluster. Otherwise, if elements are part of multiple clusters, these clusters cannot be used for a hierarchical composition. To determine the number of clusters, there are two possibilities, depending on the algorithm used. Either the number must be predefined or, like in the case of *CoClust*, the best number of clusters can be determined by an internal evaluation method. For the Media Transparency Database, we determined nine clusters. The process will be explained in Chapter 5.

Displaying only clusters and their interconnections without showing any containing element will not be helpful, since there will be no information gained from this visualization. Also, displaying every element within a cluster will be no option, since it results in the same problem as described before. The idea is now to combine the clustered elements with a cut-off approach. This means that not all elements per cluster will be displayed but the ones with the largest sums of its edge weights. The other ones will be aggregated to a new list element. Additionally, the maximum weight that can be displayed with 1px (ω_{1px}) is also restricted. It gets determined by the ratio of the sum of all edge weights ($\sum \omega$) from one set into the other one to the display height ($h_{display}$):

$$\omega_{1px} = \frac{\sum \omega}{h_{display}}$$

Below, we describe how these clusters and elements (entities) are visualized and how they can be interactively explored.

4.1 Cluster Bars

One cluster is made up of two opposing *Cluster Bars* (see Figure 4.2). The height of each *Cluster Bar* is determined by the aggregated sum of edge weights of its containing entities. These clusters are also sorted by their accumulated edge weights in descending order. The edges between *Cluster Bars* represent the aggregated connections between entities, whereas the thickness also scales with the corresponding summed edge weights. If a cluster has no edge to another cluster, it means that the underlying graph is disconnected and hence this cluster represents the disconnected subgraph of the original graph.

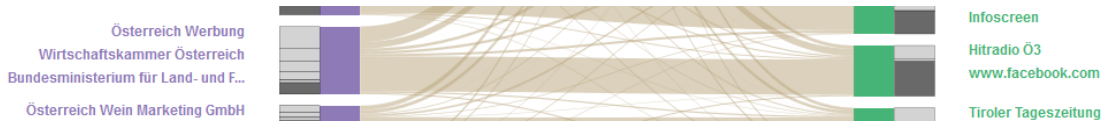


Figure 4.2: Example of a cluster consisting of two opposing *Cluster Bars* (purple and green).

For coloring the bars, we considered the qualitative color schemes suggested by Harrower and Brewer [HB03]. Those schemes were actually proposed for coloring maps with a non-numeric attribute, thus offering a good possibility to distinguish between the different attributes. However, in our visualization we also use two different sets of entities and thus can adapt the scheme for our purpose. The colors should have the same saturation and should also be colorblind safe, meaning that although the bars are geometrically separated, they should have colors that do not look the same for people suffering of color blindness. We decided on purple (RGB(143, 122, 184)) and green (RGB(70, 180, 119)) for the two sets and mongoose (RGB(184, 163, 122)) as the complementary color of these two for the edges.

Figure 4.3 illustrates how these *Cluster Bars* are connected to the underlying biadjacency matrix. The example shows a matrix with a block diagonal structure and its three biclusters. The black squares represent the connections between each row and column element (gray squares). Every cluster consists of several row and column elements, which form the *Cluster Bars*. They are colored accordingly to emphasize their usage in BiCFlows.

4.2 Stacked Bars

To show the containing entities of every *Cluster Bar*, *Stacked Bars* are introduced. Those bars are placed next to every *Cluster Bar*. Every bar thereby represents a single entity, where its summed edge weight is encoded in its height. As mentioned above, only a limited number of entities are shown. Every *Stacked Bar* that has a height smaller than 2px will be aggregated to a single darker bar (see Figure 4.4a). Consequently, the height of this darker bar is the summed height of its containing entities. The limit of 2px was used, because it is the smallest height that can be used to still be able to hover over the

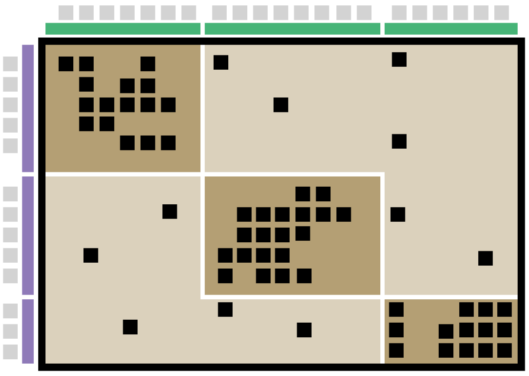


Figure 4.3: Example of a biclustered biadjacency matrix and its connection to *Cluster Bars* (purple and green).

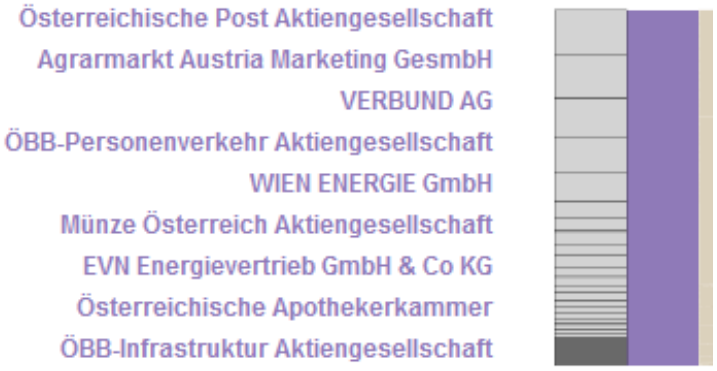
bar and get detailed information. We did not color the *Stacked Bars* the same way we did the *Cluster Bars*, but used a light gray (RGB(211,211,211)) for all entities, independently of their set affiliation. This was done because there is no gap between single entities to separate them from each other and thus it would be difficult to distinguish between each of them if the same coloring scheme as for the *Cluster Bars* would be used (see Figure 4.4b).

4.3 Labeling

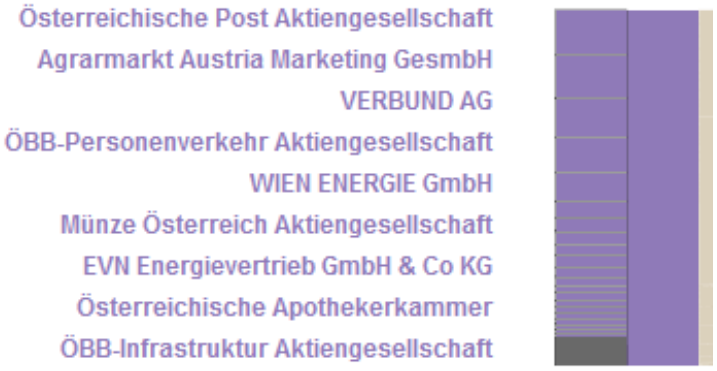
BiCFlows has a specific labeling scheme. Labels could have been put next to its corresponding stacked bar, if the height of the bar is at least as high as the label's height, which is 12px in our case. However, this will result in a very small number of labels and there will also be clusters without any label, as can be seen in Figure 4.5a. Hence, for every *Cluster Bar*, the containing entities with the largest sums of edge weights will be displayed next to it. The number of labels is limited by the height of the *Cluster Bar* (see Figure 4.5b). For aggregated stacked bars, the label of its largest entity will be displayed within squared brackets to indicate that this label does not belong to a single entity, but an aggregated group of it.

4.4 Highlighting

There are two possibilities in BiCFlows to highlight or focus on individual connections. If the interest lies in interconnections of a certain *Cluster Bar*, it is possible to hover over it. Hovering a *Cluster Bar* brings out only the other *Cluster Bars* connected with it. This means that only connections of entities in the hovered *Cluster Bar* are taken into account. The *Cluster Bars* and their *Stacked Bars* on the opposite site connected with the hovered *Cluster Bar* change their heights accordingly. Figure 4.6a shows how this



(a)



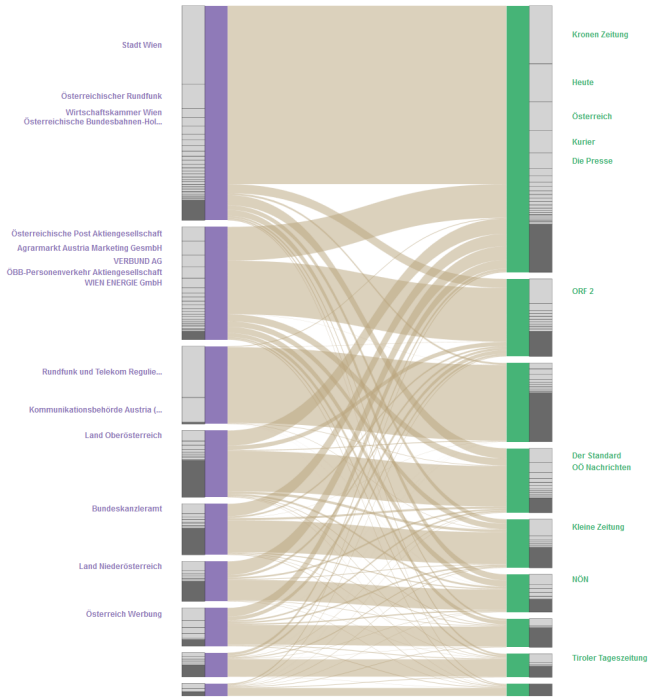
(b)

Figure 4.4: Example of *Stacked Bars*: (a) As used in BiCFlows; the light gray bars represent entities, whereas the dark gray bar consists of all other aggregated entities within this *Cluster Bar*. (b) Example if *Stacked Bars* were colored the same way as *Cluster Bars*.

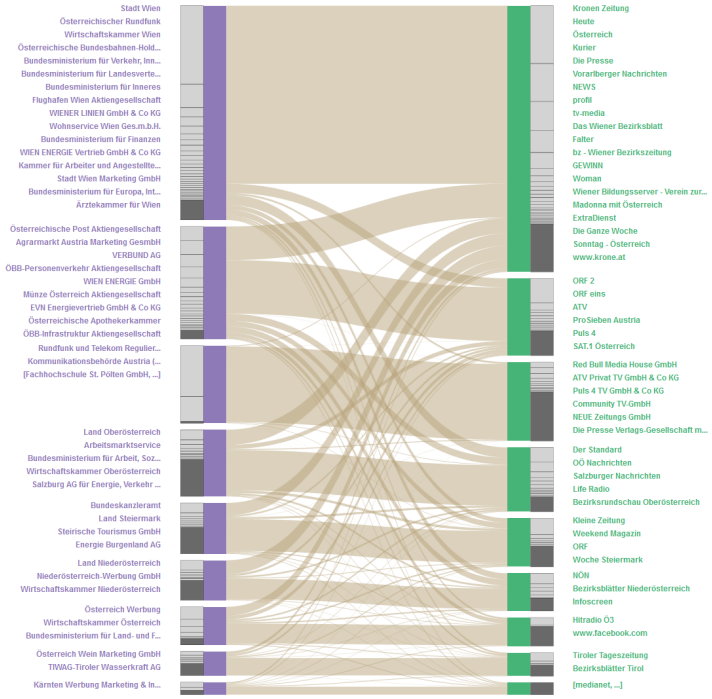
fanned out representation looks like. All other edges are faded out to keep the focus on the hovered *Cluster Bar* and its connections.

If the interest lies in a single entity, represented as a *Stacked Bar*, it is possible to hover over it as well. Hovering a single entity shows its connections to every other entity on the opposite site. The connections are illustrated as red edges, where the thickness of the edge scales with its weight. Additionally, all labels corresponding to the connected entities will be highlighted in red as well. Figure 4.6b shows an example where *Land Oberösterreich* is hovered. Like with highlighting *Cluster Bars*, all other connections are faded out to keep the focus on the hovered entity.

In both cases it is also possible to lock the highlighting by clicking on the *Cluster Bar* or *Stacked Bar*, respectively. This allows the user to examine single edges. Figure 4.7 shows an example where *Stadt Wien* was locked and then a single edge is inspected. Again, hovering over an edge fades out to other connections.

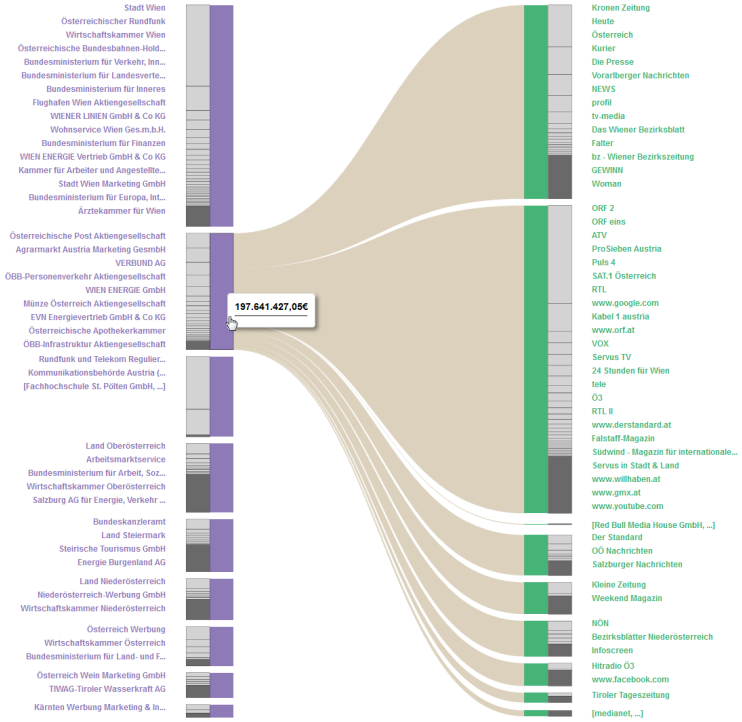


(a)

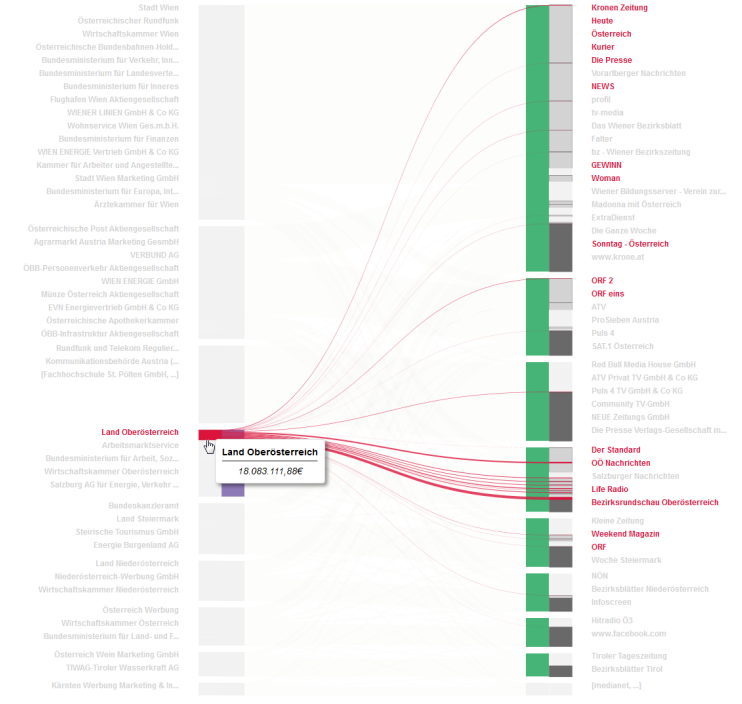


(b)

Figure 4.5: Labeling: (a) label per stacked bar, if bar height is greater than label height (12px); (b) maximum labels per cluster bar.



(a)



(b)

Figure 4.6: Hovering over: (a) a *Cluster Bar* and (b) a *Stacked Bar*

Figure 4.7: Example of hovering over an edge after locking the highlighting of a *Stacked Bar*.

4.5 Navigation

To get more information about entities within a cluster and especially about the aggregated stacked bar, it is possible to move one level down in the bicluster hierarchy by double-clicking on a *Cluster Bar*. By doing so, the entities contained by this cluster get biclustered again and result in a similar view like the initial one (see Figure 4.8).

The difference is that entities that were part of other clusters in the previous view, but are connected to entities in the current cluster are aggregated to new *Cluster Bars* with yellowish color (RGB(175, 192, 23)). To better understand this subclustering process, Figure 4.9 illustrates the procedure. The red-bordered cluster is the one that was double-clicked. During the biclustering process its rows and columns get rearranged to find new clusters. These clusters form the base for the new *Cluster Bars* and its entities. The yellowish *Cluster Bars* however, consist of entities from the previous cluster that shared a connection to the double-clicked cluster.

The reason for this is the following: If a cluster gets selected, the main interest consists of the entities within this cluster and its interconnections. To omit no information or give the false impression that entities of the selected cluster only share intrinsic connections, the entities of the other clusters will be aggregated in this way. It also indicates how enclosed a cluster is. The smaller the yellowish *Cluster Bars* are, the fewer connections are shared with entities outside of this cluster.

Every new cluster can be further explored the same way as described above. This leads to levels where fewer and fewer entities are present. There are two cases where further subclustering does not change the output. In the first case, the underlying biadjacency matrix that will be clustered becomes 1-dimensional, meaning that it only consists of one row or one column. This results in only one cluster where one of the two *Cluster Bars* consist of only one entity. In the other case, every row element is connected to every column element in the underlying biadjacency matrix. This results also in only one cluster, but here the *Cluster Bars* consist of every row and column element, respectively.

4.6 Context Bars

When going down the cluster hierarchy through subclustering as described above, there is also the need to return to previous hierarchy levels. To navigate up and also to keep

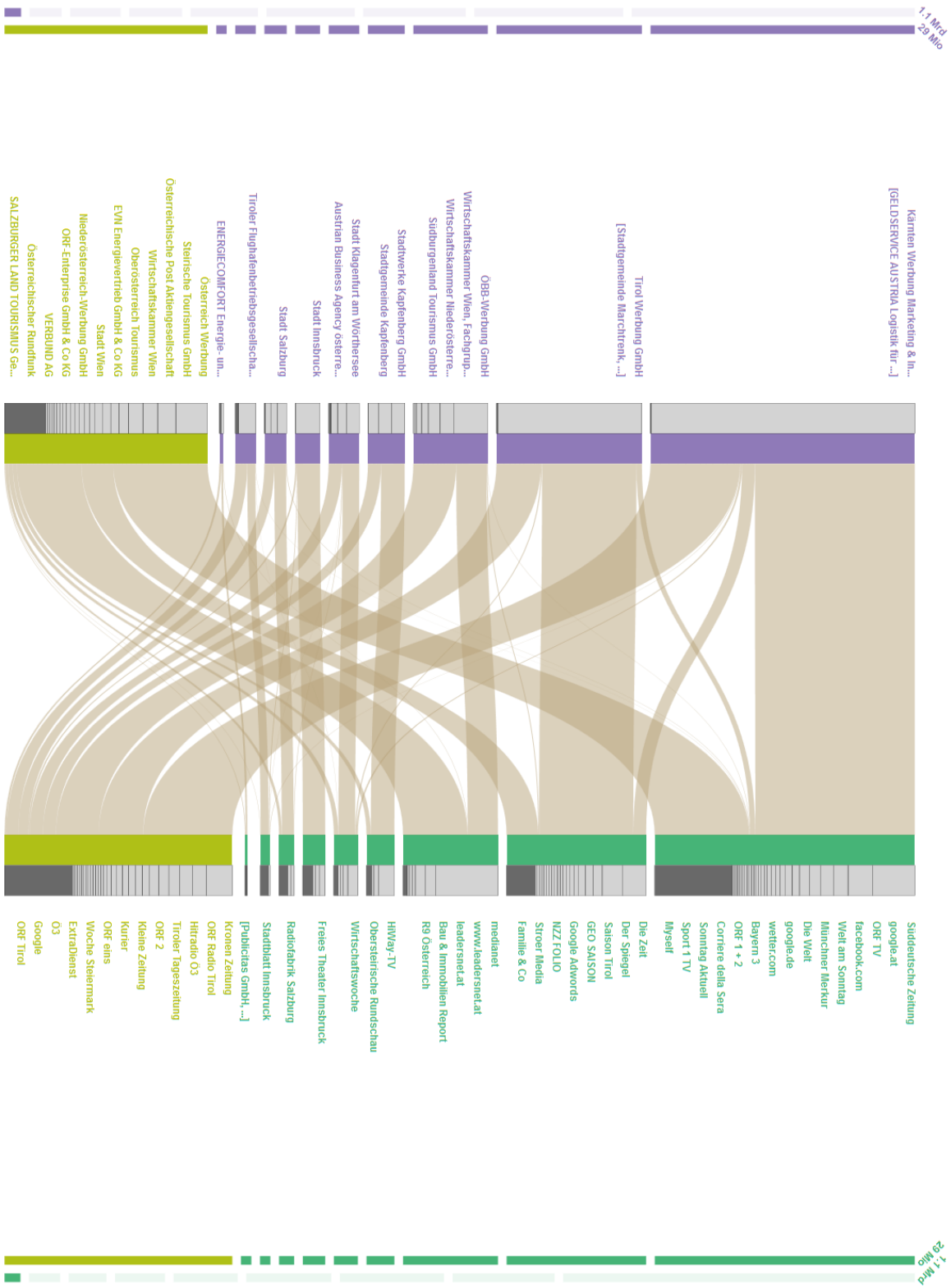


Figure 4.8: Example of one level deeper in cluster hierarchy.

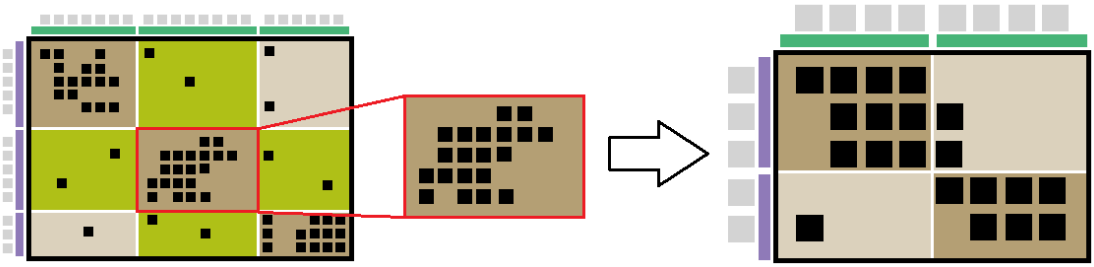


Figure 4.9: Example of the subclustering procedure resulting in a new biclustered cluster.

track of the current level within the hierarchy, each side of the two sets of lists provides so-called *Context Bars* (see Figure 4.1g). They can be understood as a minified version of the *Cluster Bars*. Every time a *Cluster Bar* is double-clicked, thus navigated one level down in the hierarchy, a new *Context Bar* will be added to each side. The *Cluster Bar* that was double-clicked will be highlighted within the last *Context Bar* (see Figure 4.8). This allows for tracing back one's navigation within the hierarchy. At the same time, these *Context Bars* serve as possibility to navigate up again. By clicking a *Context Bar*, the level of this specific cluster hierarchy will be loaded. This does not mean that one can navigate up one level at a time, but also several levels can be traversed at the same time or even completely up to reach the initial view.

4.7 Additional Views

To follow Shneiderman's information seeking mantra mentioned in Section 3.3, BiCFlows offers some additional views to further explore and filter the data. Moreover, detailed information is provided in the form of tooltips as can be seen in Figures 4.6 and 4.7. Whenever a certain element, like bar or edge, is hovered, its properties will be displayed.

4.7.1 Bar Charts

If the data contains additional attributes, BiCFlows offers two bar charts where these attributes can be displayed (see Figure 4.1a and b). The edge weights will be aggregated per bar. The bar charts can be used to compare and filter by the assigned attribute. In case of the Media Transparency Database, the three paragraphs that form the legal basis of its disclosures and the quarters are used. Multiple bars can be selected to filter the data, whereas selected bars are colored differently.

4.7.2 Tables

In order to directly search for well-known entities or to sort entities by their accumulated edge weights, BiCFlows offers two tables (see Figure 4.1c and d). The tables can be sorted in ascending or descending order by name and weight. If an entity is selected in

the table, it will be highlighted in the main view. If the selected entity is not present as an independent *Stacked Bar* in the main view, but is part of an aggregated bar because its weight sum is too small, the aggregated bar will be highlighted.

CHAPTER 5

Implementation

BiCFlows was implemented using a client-server infrastructure. This architecture was chosen to separate the computationally more intensive biclustering procedure from the user interface on the client side. In the following sections, we will first deal with the server side, explain how the biclustering is handled in *BiCFlows*, and then focus on the client side, as well as their server-client-interactions.

5.1 Server

The server backend was implemented with *Python* (version 3.6.1) [Fou18] using *Flask* (version 0.12.2) [Ron18], which is a micro web-framework used to easily set up web-applications. This framework was used, because it offers the possibility to quickly set up a suitable application with routing functionality without having to deal with more bloated and sophisticated web-frameworks like *Django*. Since we are also dealing with a large data set with several thousand entries that need to be aggregated and manipulated, we use the *Python* library *Numpy* (version 1.13.1) [Oli17], which can process large, multi-dimensional arrays and matrices faster than *Python's* built-in data structures [WCV11]. Moreover, *Python* offers – in comparison to *JavaScript* – access to powerful libraries, like *NetworkX* (version 1.11) [HSS17], which we used to generate the biadjacency matrix from the raw data.

If biclustering was done in the browser, this could lead to delays due to inefficiencies in *JavaScript* and limited computing power on the client-side, which will in turn have a negative impact on the exploration experience. That is why we handle the computation on the server using *Python*, where the following bicluster implementations exist:
Ailem et al. [ARN15] made their biclustering approach publicly available as a *Python* package *CoClust* (version 0.2.1) [FSM17]. Moreover, the *biclustering module* [Bic17] as part of the *scikit-learn* (version 0.18.2) [PVG+11] machine learning package for *Python* offers two implementations that assume different underlying bicluster structures (see

Section 2.3). The one by Dhillon [Dhi01] assumes a block diagonal structure and the other one by Kluger et al. [KBCG03] assumes a checkerboard structure.

In our implementation, we used the algorithm proposed by *Ailem et al.* [ARN15] since it can – compared to Dhillon's approach – handle weighted biadjacency matrices and assumes a block diagonal structure that we need for our data, because we are looking for rows and columns in our biadjacency matrix that are part of only a single bicluster.

CoClust also offers the possibility to determine the best number of clusters by using an internal evaluation method. This method takes a biadjacency matrix and a range of numbers as input and returns the number where the graph modularity has its maximum. Figure 5.1 shows that the highest modularity for the Media Transparency Database in a range between two and twelve clusters was found for nine clusters. On a system with an Intel i7-4790K CPU with 4GHz and 8GB RAM, this method takes five seconds to evaluate the best number of clusters. While it would be acceptable to run this method once on startup for the whole dataset, it cannot be called on-the-fly for every subset that results through subclustering. This would make a responsive interaction impossible. That is why we decided to calculate the number of clusters beforehand and then use it as fixed value. If a completely different dataset will be used, the best number of clusters has to be recalculated.

Another aspect to mention is that *CoClust* uses random seed values by default to determine the biclusters and its entities. In Section 2.3 we explained that the algorithm initially assigns random cluster memberships to the entities before it iteratively improves the clustering. The seed values can thereby be understood as these initially random assignments. This results in different assignments after every call. However, it offers the possibility to use fixed seed values, which we did, to reach a consistent biclustering.

5.2 Client

The client was implemented using *JavaScript*, the *D3.js* [BOH11] library for visualization, and *Viz* [Viz17], a collection of visualization layouts, where the bipartite layout serves as basis for our implementation. *D3.js* offers the possibility to easily create individual visualizations using HTML, SVG, and CSS. It follows a data-driven approach to manipulate the Document Object Model (DOM) and is thus very flexible [Bos17].
Furthermore, *DataTables* [Spr17], a table plug-in for *jQuery* [jF17] was used to show all the single entries of media organizations and legal entities in two separate tables. Besides providing the possibilities to search and sort the table's content, it offers a virtual rendering plug-in. This plug-in makes it possible to display large data sets, because only the visible entries and not the whole data set are rendered.
To quickly update these tables and the bar charts if the user filters the data by quarter or legal basis, *Crossfilter* [Squ17] was used. *Crossfilter* is a *JavaScript* library that is able to filter data on multiple dimensions very quickly and is thus ideal for the application in a coordinated views setup.

Figure 5.1: The output of the internal evaluation method of *CoClust* shows the highest modularity for nine clusters.

it is also able to display large data set, by using a

5.3 Server-Client-Interaction

Since the biclustering process is completely outsourced to the server, the client has to call server-side methods to get the updated data, if a user interaction has occurred. The procedures in Figure 5.2 show, how client and server interact with each other.

Figure 5.2a deals with the initial routine when the system is started. On start-up, the client calls the server's method to get the data, which parses the data that is stored in a .csv file and sends it back to the client. The client then uses this data to fill the tables and bar charts. Additionally, it calculates the number of clusters based on the height of the current browser window, where the visualization will be displayed and sends this number back to the server, where it is stored for future cluster-calls. Based on an average monitor with an aspect ratio of 16:9 and a Full HD resolution of 1920×1080 pixels, we defined nine clusters as a default value for a display height of 1080 pixels and above. For smaller resolutions, we used a function that linearly maps the interval between 400 and 1080 pixels to the interval between two and nine clusters. This allows us to automatically scale the number of clusters based on the display height, since too many clusters on a small display will crowd the visualization too much. Afterwards, the client calls the server's method to get the biclustered data, which will be calculated on the server-side using the previously received number of clusters and then sent back to the client where the clustered data will be displayed. Once the data was sent to the client, the whole

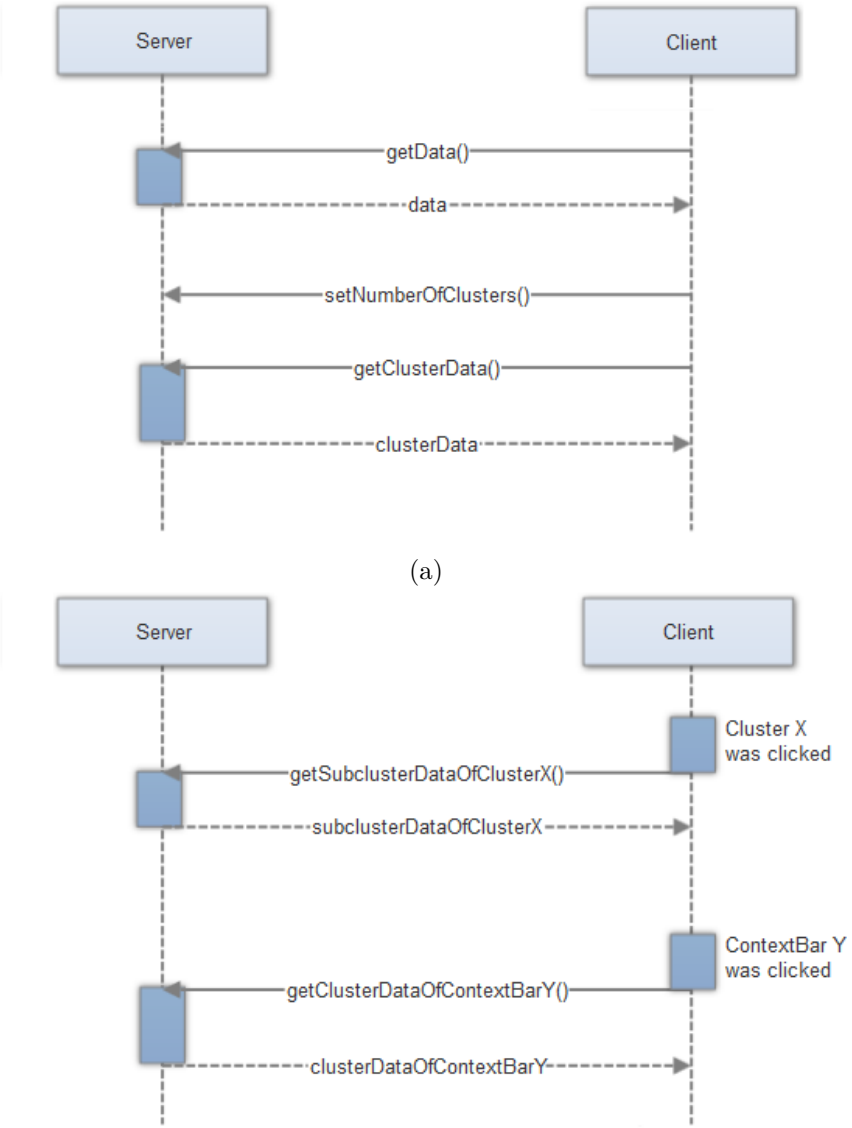


Figure 5.2: Client-Server-Interaction: (a) initial routine, (b) subclustering.

visualization part including hovering, expanding, and highlighting effects is done at the client-side. For this purposes no interaction with the server is needed.

Interaction is only needed if clusters must be newly determined. Thus, the client listens for click events on *Cluster Bars* or *Context Bars*. Figure 5.2b shows that after a cluster is clicked, the server calculates the subclusters of this cluster and sends the data back to the client. Likewise, if the user wants to navigate a level up in the cluster hierarchy and clicks a *Context Bar*, the data is calculated and sent back to the client.

CHAPTER 6

Case Studies

In this chapter, we first introduce the "Cut-Off approach" that was used as comparative user interface in our user study and then we present some use cases. Besides the Media Transparency Database, we will have a look at the possibility of using BiCFlows for publication data as well as for movie data.

6.1 Cut-Off Approach

The Cut-Off approach was implemented to serve as a comparison visualization for our evaluation (see Chapter 7). Basically, it is similar to the visualization shown in Figure 1.4, with the exception that the radial layout has been replaced by a list to be more comparable to BiCFlows. Figure 6.1 shows the initial view of the cut-off approach. In this approach, only entities with the largest sums will be displayed and the rest will be aggregated within a new bar. Like in BiCFlows, the sums are encoded in the bar heights. The aggregation starts at the point where the bar height would be smaller than its label height. In our implementation, we used a label height of 12px. Apart from that, we used a very similar visual encoding to BiCFlows to establish a good basis for comparison. We used the same color scheme, the same layout, and also the same label height. The only difference is that the colored bars, which were introduced in BiCFlows as *Cluster Bars* (see Section 4.1), now here serve as bars representing single entities, because the Cut-Off approach does not use clustering. We also kept the gap between the bars to visually separate entities better from each other.

The highlighting also works similarly to the one in BiCFlows. Initially, the edges of all visible entities are displayed. Hovering over a certain entity brings out only the connections of this specific entity. All other edges are faded out, to better trace the connections of the hovered entity (see Figure 6.2a). Similarly, if only a single edge is hovered, this specific edge will be displayed and the others are faded out (see Figure 6.2b). To explore the data, it is possible to select an entity in the main view or the lists. As a

result, the connections of this entity will be shown as illustrated in Figure 6.3. It is also possible to select multiple entities to compare them with each other.

6.2 Media Transparency Database

Our main motivation in developing BiCFlows was the lack of appropriate tools to explore large bipartite data like the Media Transparency Database. We already explained the general design choices in Chapter 4, but here we will discuss data specific decisions. In our design, we offer the possibility to aggregate and filter data by two additional attributes (see Section 4.7), which is also provided in the Cut-Off approach. In case of the Media Transparency Database, these two attributes are the legal bases and the quarters. As mentioned in Section 1.1, the Media Transparency Database has three paragraphs that form the legal basis of its disclosures. Figure 6.4a shows these paragraphs as bar chart, where the sums of all entities are aggregated for each paragraph respectively. This allows the user to compare the legal bases with each other and also to filter the data accordingly. The same principle applies for quarters, where transaction sums are aggregated for each quarter. Thus, time periods can be compared with each other and also be filtered by them (see Figure 6.4b).

Answering predefined questions, like which entity spent or received the most or less in a certain year or period is possible in both, BiCFlows and the Cut-Off approach. However, the main advantage of BiCFlows lies in its capability to support free exploration, thus finding unexpected information without a specific problem in mind. We examined this exploration process in our user study and some of the findings that users made during their exploration will be discussed in Section 7.6.

If we compare the Cut-Off approach to BiCFlows in terms of exploration options, we see that the only possibility to explore the data here is by interacting with the initially visible entities or by scrolling through the tables. However, most users will be looking for entities that they are familiar with in the tables, thus not finding anything new. A specific question, like *How much money did entity X receive?* could in fact be answered faster by looking at the tables. However, it would be also interesting to find similar entities that receive money from the same entities. We try to illustrate this in the following scenario:

A user is interested in her local newspaper *Salzburger Nachrichten* and explores it with BiCFlows. Since *Salzburger Nachrichten* is already visible in the initial view, she can open up the cluster containing it and go even further down the cluster hierarchy a few more times. During her exploration, she discovers that *Salzburger Woche* and *Salzburger Fenster*, which she had never heard of, receive money from the same legal entities as *Salzburger Nachrichten* does (see Figure 6.5).

In the Cut-Off approach however, *Salzburger Nachrichten* is not visible in the initial view, because its total receiving sum is too small to be displayed and consequently gets aggregated. Therefore, she has to search for it in the table and select it from there. By selecting it, only legal entities that are advertising in *Salzburger Nachrichten* are displayed

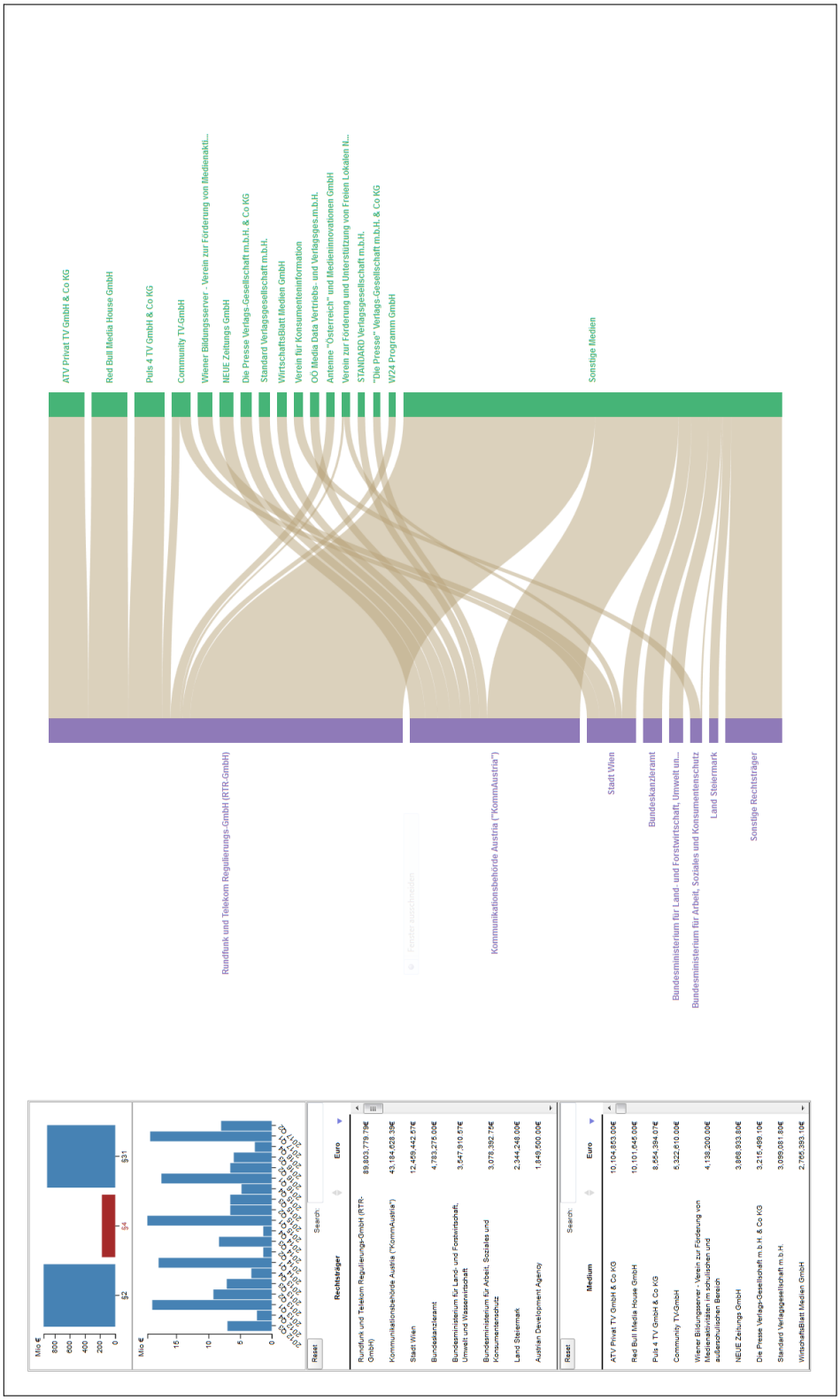
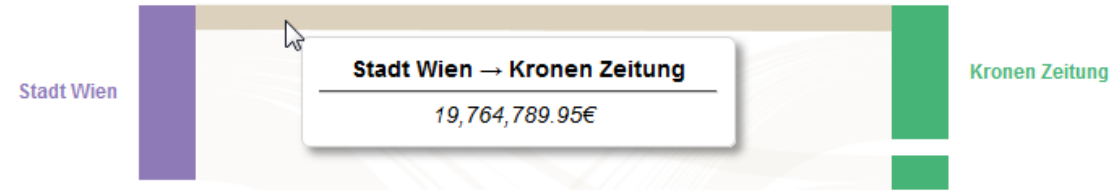


Figure 6.1: The implemented Cut-Off approach.

(a)



(b)

Figure 6.2: Examples of hovering in the Cut-Off approach: (a) over an entity, and (b) over an edge.

but no other media organizations (see Figure 6.3). This makes it nearly impossible to find similar media organizations.

If *Salzburger Nachrichten* would not have been visible initially in BiCFlows, the user could have also searched for it in the table. After selection, either the entity or - if it is aggregated - the dark gray aggregation bar containing it would have been highlighted. From then on, the exploration process would be the same as described above.

The fact that in the previous example *Salzburger Nachrichten* was visible in BiCFlows in the initial view is also conditioned by the labeling scheme described in Section 4.3. Compared to the Cut-Off approach where 30 labels are visible initially, BiCFlows shows 95 labels. This also influences other use cases, for example, if a user is interested in finding all media organization where *VERBUND AG* is advertising in. Figure 6.6 shows that there are initially more connected media organizations visible in BiCFlows (38) than in the Cut-Off approach (12). This is also the case after going into detail in both user interfaces, thus selecting *VERBUND AG* in the Cut-Off approach and going down the cluster hierarchy in BiCFlows to a level where only *VERBUND AG* is visible on the left side. In both cases the number of labels increases, but BiCFlows shows still more (53) than the Cut-Off approach (25) (see Figure 6.7).

Nevertheless, the Cut-Off approach benefits from its capability of comparing two or more entities with each other. Figure 6.8 shows an example where *ORF eins* and *ORF 2* are selected. This enables the user to compare these two entities and their cash flows directly, which is not possible in the current version of BiCFlows. However, it would be

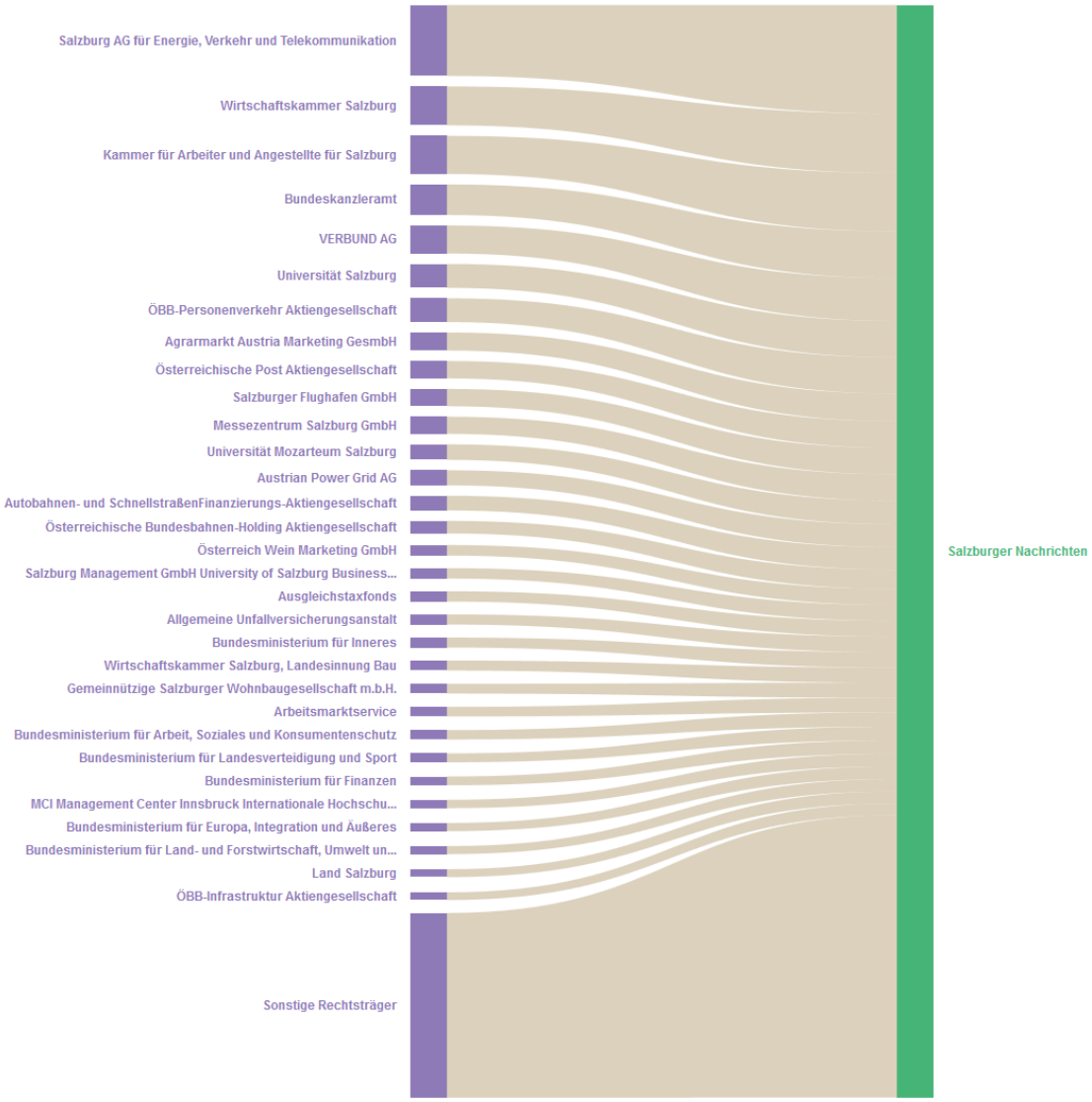


Figure 6.3: Cut-off approach with selected entity *Salzburger Nachrichten*.

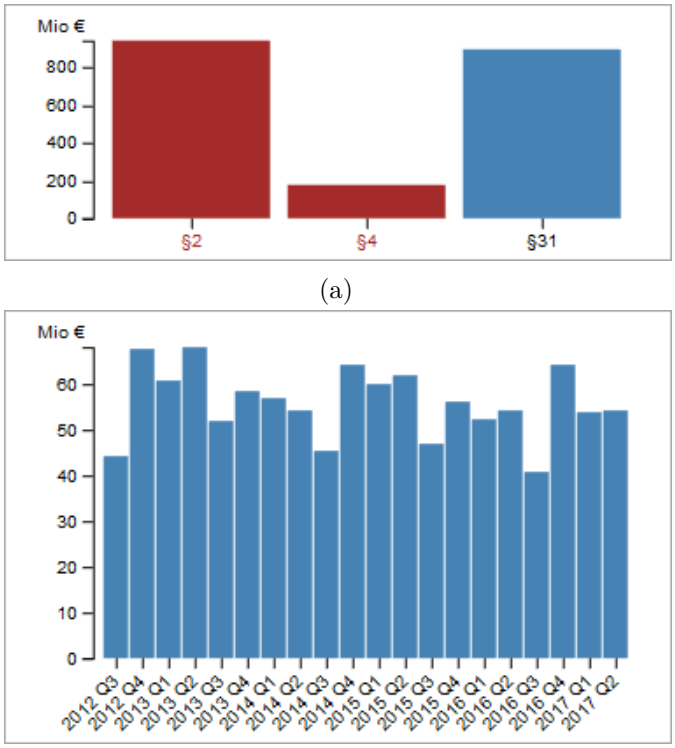


Figure 6.4: Two additional views in BiCFlows showing aggregated sums per attribute: (a) legal bases with §2 and §4 selected, and (b) quarters.

conceptually possible and definitely a good point of improvement for future works.

6.3 Publication Database

In 2014, Isenberg et al. [IHK+17] started to collect data about IEEE Visualization Publications. Currently, the dataset covers all publications from 1990 to 2015 of all constituent conferences (InfoVis, VAST, SciVis and Vis) and is available at their website *vispubdata.org* [IHK+18]. The information of each paper includes its title, authors, year, IEEE terms, author keywords, DOI, etc. The idea was to make the dataset publicly available to other researchers and encourage them to develop visualizations for exploring this data. The following three visualizations tools are examples they developed themselves. *CiteVis2* [IHK+17] is focused on determining citation counts per year and conference and also on finding specific cross-referenced papers. *CiteMatrix* [IHK+17] is a matrix-like visualization that shows how often papers in different conferences cite each other as well as the citation trend over time. Finally, *VISLists* [IHK+17] is a visualization that can be used to find out more about an author's publication output across all conferences including their most frequent co-authors (see Figure 6.9).

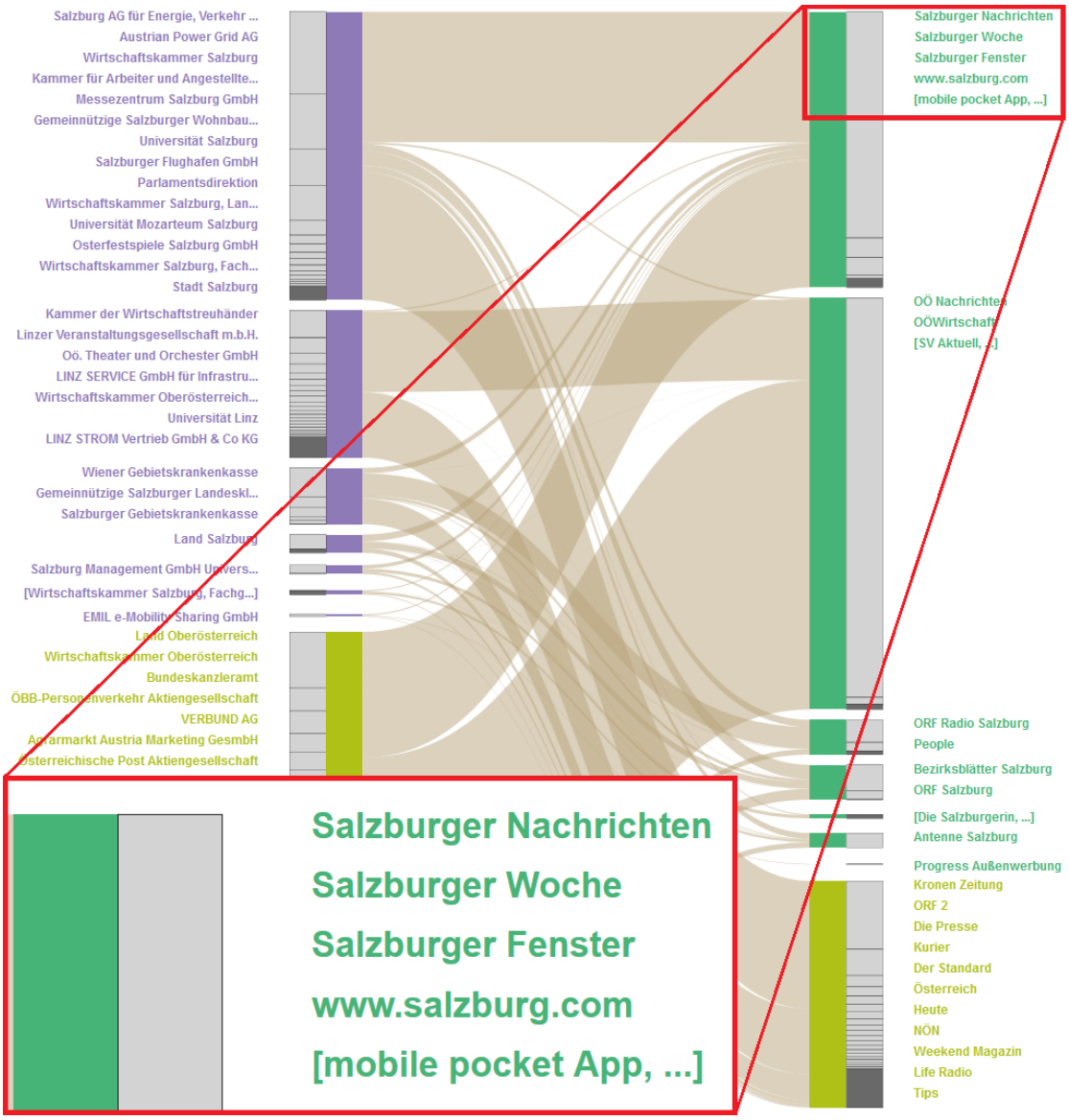


Figure 6.5: Looking for similar media organizations like *Salzburger Nachrichten* in BiCFlows. The red rectangle marks media organizations that receive money from the same legal entities as *Salzburger Nachrichten* does.

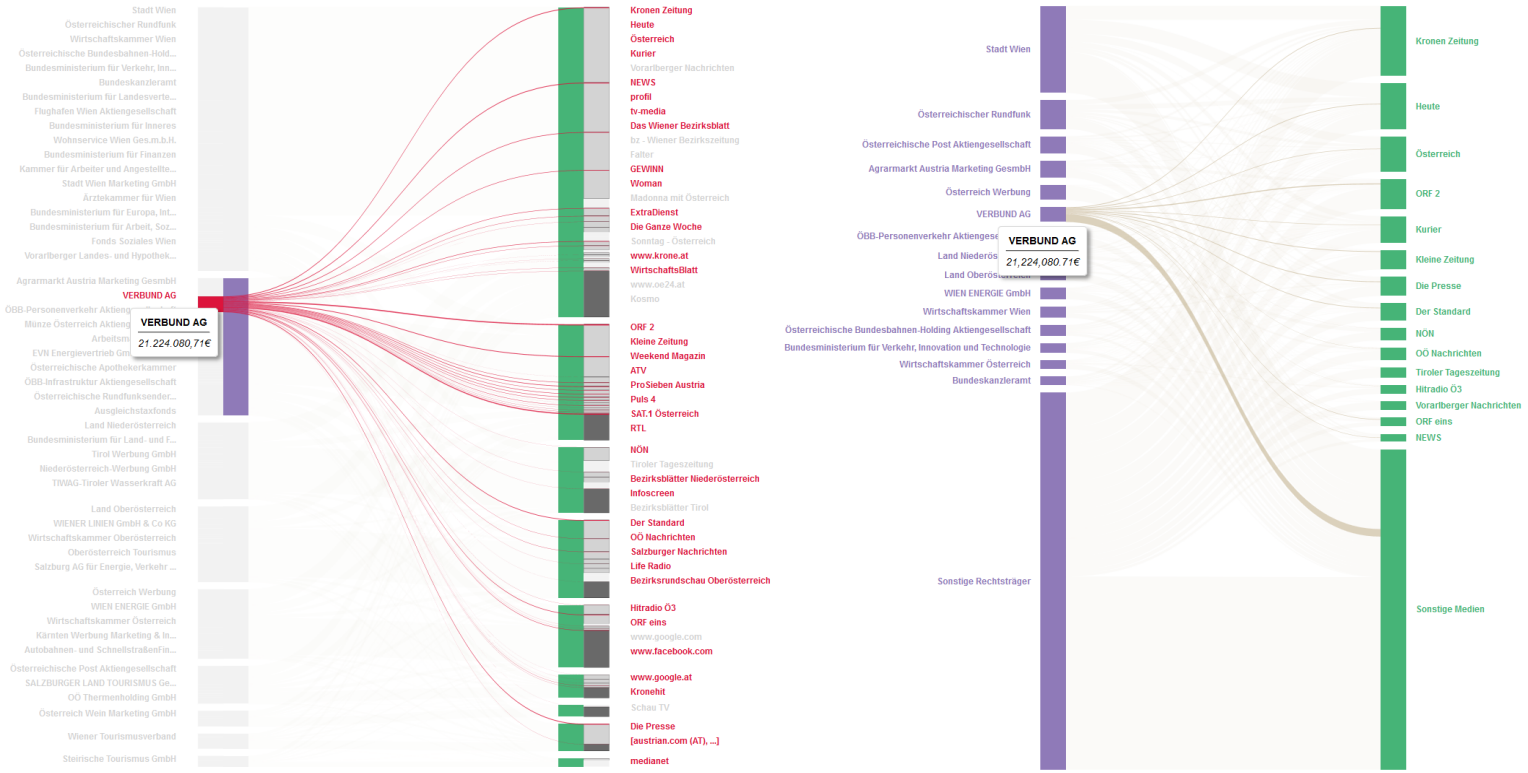


Figure 6.6: Comparison of visible connected labels between (a) BiCFlows and (b) the Cut-Off approach with selected legal entity *VERBUND AG* in the initial view.

(a) (b)

Figure 6.7: Comparison of visible connected labels between (a) BiCFlows and (b) the Cut-Off approach with selected legal entity *VERBUND AG* in the detail view.

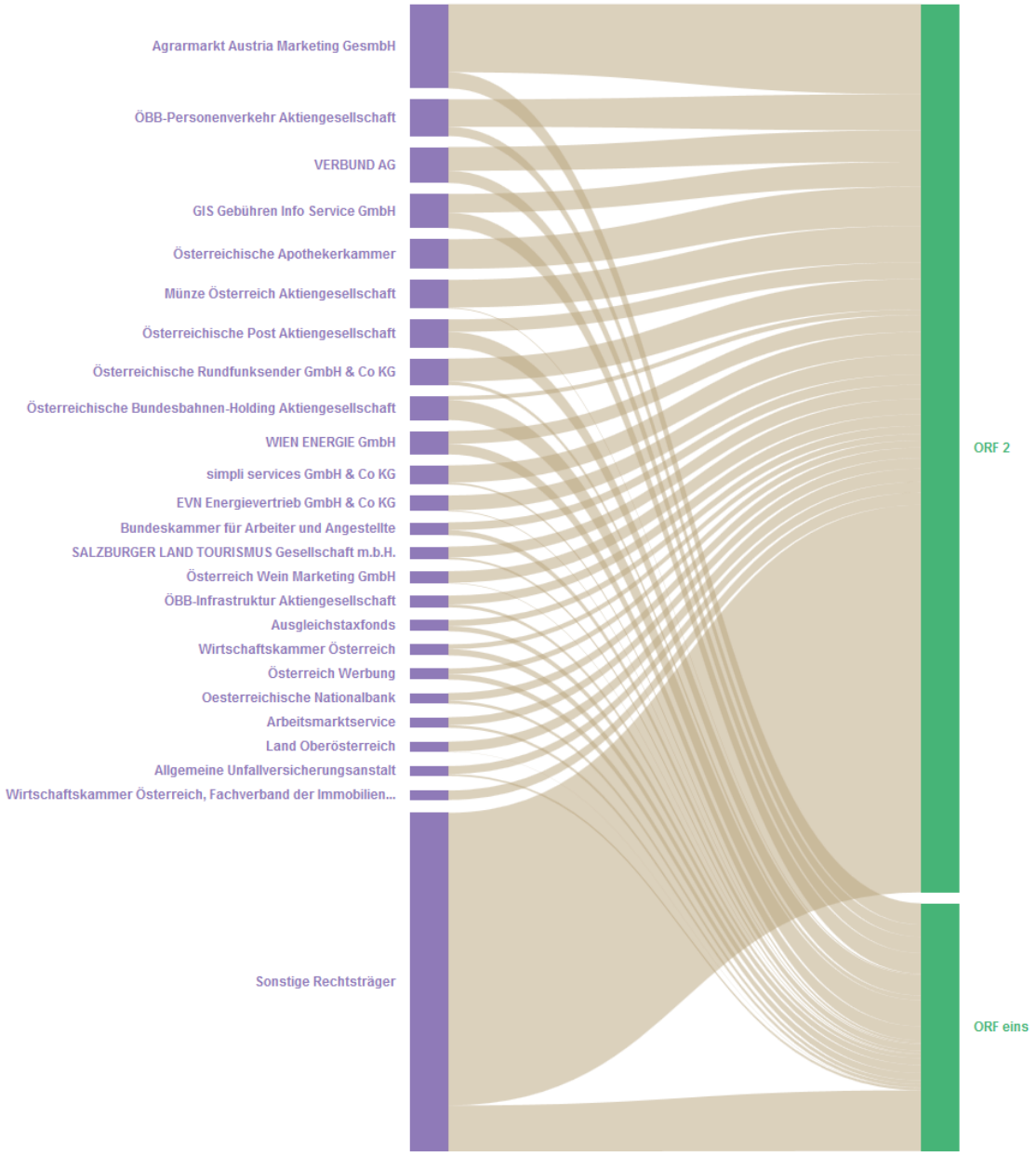


Figure 6.8: Cut-off approach with two selected entities *ORF eins* and *ORF 2*.

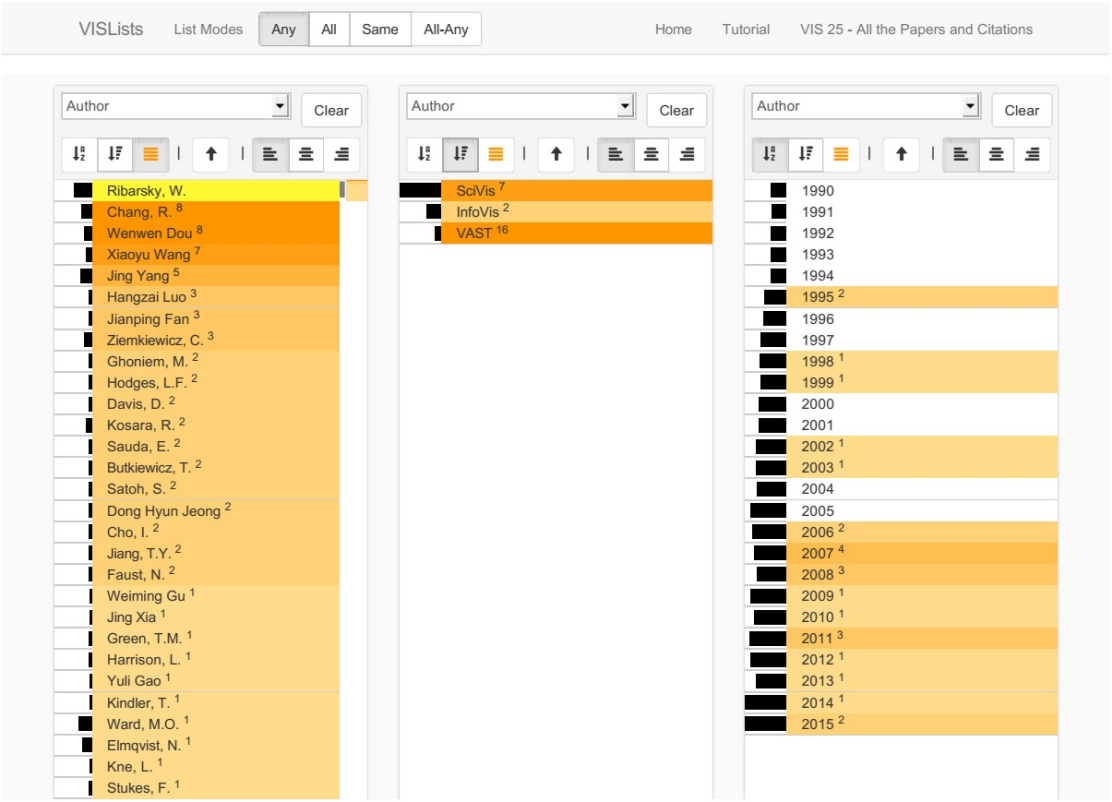


Figure 6.9: *VISLists* by Isenberg et al. [IHK+17] showing the co-authorship of author *Ribarsky*.

Isenberg et al. [IHK+17] also offer the possibility to search for author keywords at the website of *KeyVis* [IIS+18]. At the website, one can search for a specific keyword and it shows the papers that used this keyword, other keywords that co-occurred with the searched one and which higher-level topic it belongs to. Though this website delivers good result when looking for explicit keywords, it is not very exploratory. That is why we decided to use the publication dataset with BiCFlows. For this purpose we adapted the data to our needs. We were interested in the relation between authors and keywords. For keywords we did not use author keywords, but IEEE terms since they are more coherent and follow the IEEE taxonomy [IEE18]. As additional information, we used the publication's year and the conference's name, to offer the possibility to filter by these attributes through the same charts as presented in Figure 6.4. The final dataset consists of 4976 authors and 2120 terms. We determined seven as the best number of clusters with the method described in Chapter 5, though the modularity is generally low for this dataset (see Figure 6.10).

Figure 6.11 shows this dataset visualized using BiCFlows. A notable aspect is the fact that there are more stacked bars visible on the right side than on the left side. This

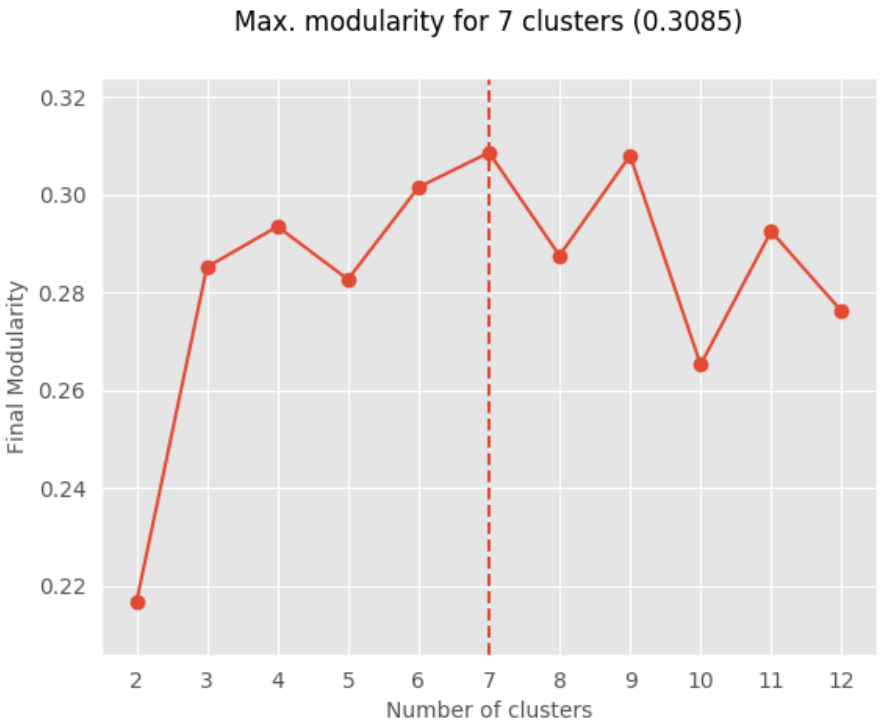


Figure 6.10: The output of the internal evaluation method of *CoClust* for the publication dataset shows the highest modularity for seven clusters.

is because some IEEE terms are used far more often on average than others like *data visualization*, *visualization*, *rendering*, or *computer graphics*, while on the other side there are no authors who use that much more IEEE terms than others do. Another interesting aspect is that, through the clustering approach, authors and keywords are now grouped together offering the possibility of revealing co-authorships or groups of people using similar keywords. This can be useful for researchers looking for possible future co-authors. We can also compare BiCFlows here with *VisList* by Isenberg et al. [IHK+17], which offers an overview of co-authorships as well. We can take the example in Figure 6.9, where the author *Ribarsky* is selected and look for authors in the same group as *Ribarsky* in BiCFlows. Figure 6.12 shows the cluster containing *Ribarsky* in BiCFlows. When comparing the authors, we see that all of them are also listed as co-authors in *VisList*. The different order is due to the fact that *VisList* is based on mutual publications, whereas BiCFlows uses common keywords. Nevertheless, this example shows that BiCFlows can in fact be used to find co-authorships.

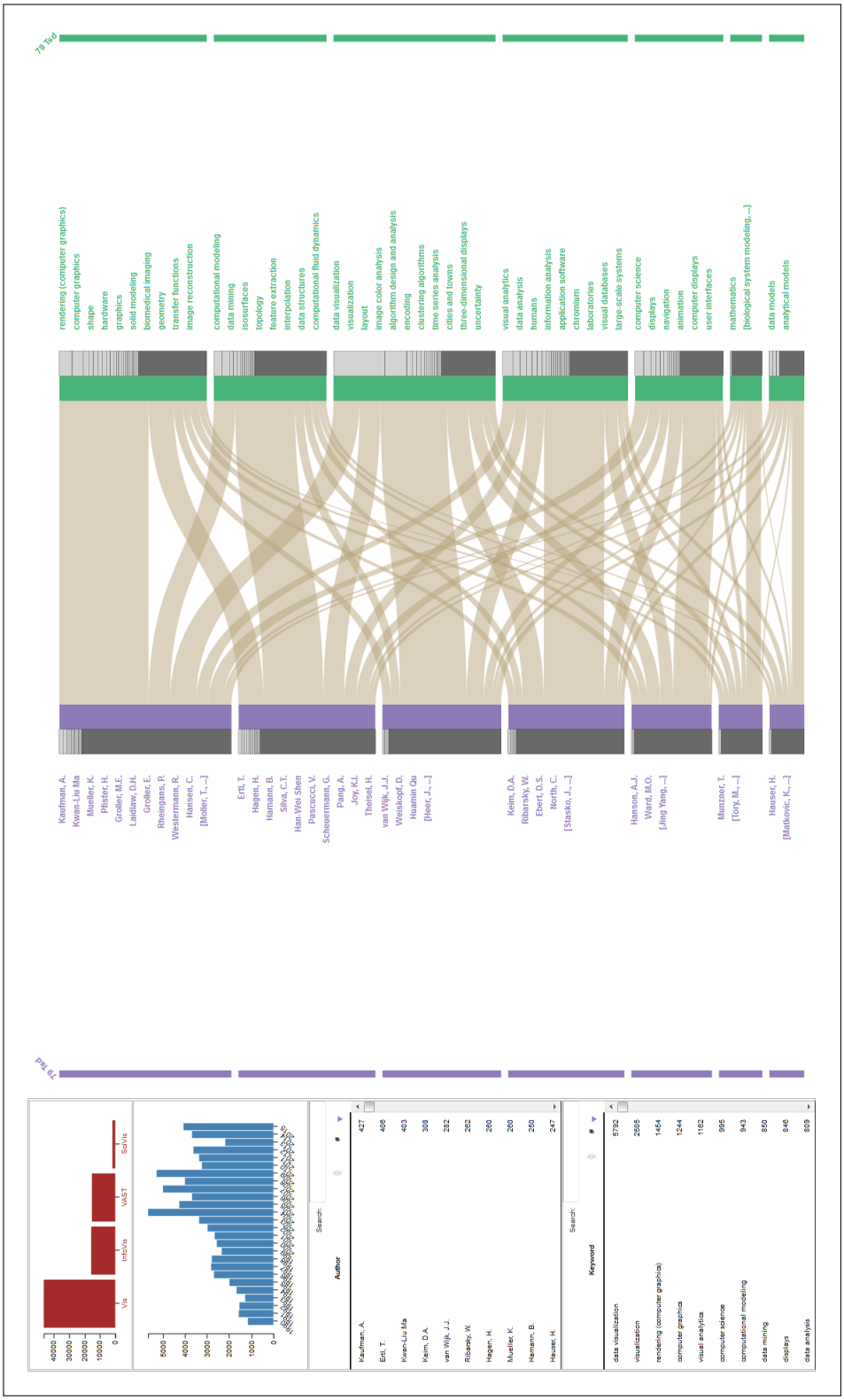


Figure 6.11: Author and keyword relation visualized using BiCFlows.

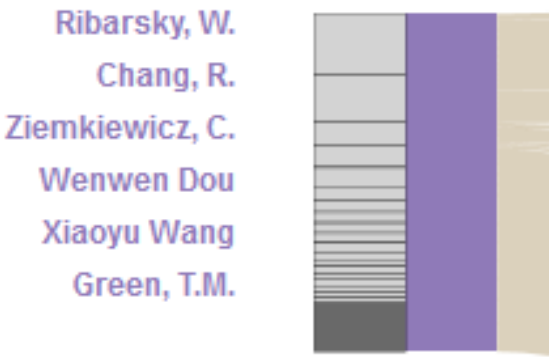


Figure 6.12: Cluster of authors containing *Ribarsky* in BiCFlows.

6.4 Movie Database

Another database that was also used as test dataset in our user study (see Section 7.2) consists of movies and viewers. Their relation is expressed through the money a viewer spent on a particular movie. As additional attributes, the genre of the movie and the year it was watched were provided. The dataset was created artificially by using a list of movies provided by the *ggplot2movies* [Wic17] package of *RStudio* in combination with a list of randomly generated names from the website [App17] of Joe Apple. The finally produced dataset consists of 943 viewers and 1611 movies limited to three genres and four quarters. For this dataset, we determined four clusters with a modularity of around 0.27. Figure 6.13 shows this dataset visualized using BiCFlows.
Although the dataset was created artificially, it has a real background. Current movie-streaming-websites may have a similar database structure, but unfortunately make their data not publicly available for research purposes. When exploring the data with BiCFlows, users may find groups of people with similar movie taste or discover unknown movies that match their favorite genres.

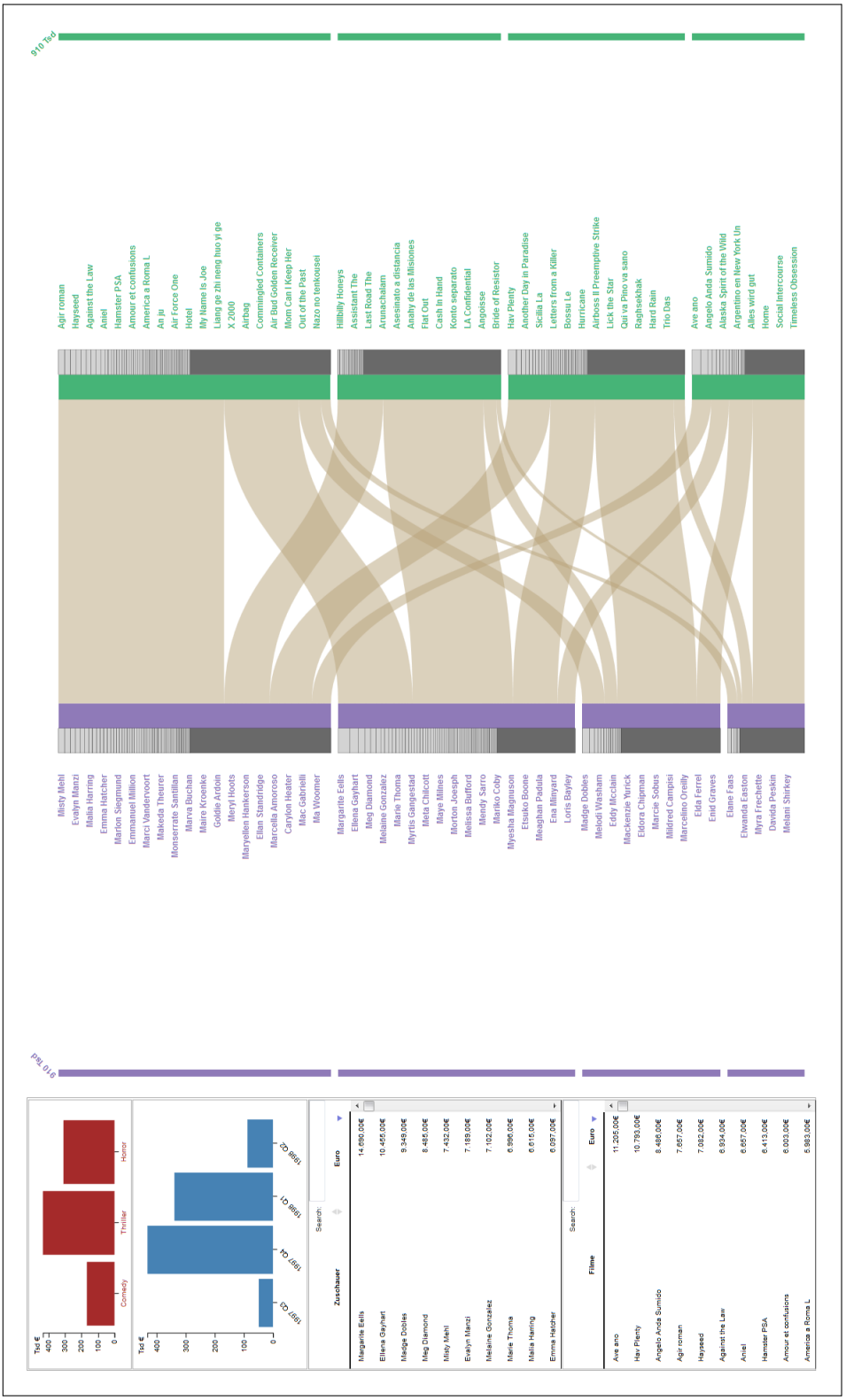


Figure 6.13: Viewer and movie relation visualized using BiCFlows.

CHAPTER 7

User Study

Basically, there are two types of evaluation methods, namely *quantitative* and *qualitative* ones [Car08]. In quantitative evaluations, users have to perform predefined tasks where the accuracy and the task-completion-time are typically evaluated. Amar et al. [AES05] collected around 200 questions for analyzing five different sets of data from different domains. They grouped them to categories covering similar questions. On its basis, they defined ten categories of low-level analytical tasks:

- **Value retrieving:** Finding a certain value for a specific case.
- **Filter:** Finding values that satisfy predefined conditions.
- **Value deriving:** Computing aggregations of given data, like mean or median.
- **Extrema:** Finding minimum or maximum of given data objects.
- **Sort:** Ordering data objects by their attributes.
- **Range:** Determining a range, where for a given attribute entities lie within.
- **Distribution:** Characterizing the distribution of data objects for a certain attribute.
- **Anomalies:** Identifying unexpected data values, like outliers in a distribution.
- **Cluster:** Finding groups of similar data objects.
- **Correlate:** Determining relationships between two attributes of data objects.

The metrics measured for completing tasks of these categories can then be used to compare different visualizations with each other. Many evaluations that used these empirical methods exist [CY00] and they are still used [HW12, FFHW15].

The other type of evaluations are qualitative evaluations. The key principle in these evaluations is that they omit predefined tasks and gain a deeper understanding of the users' exploration process. One example is a concept proposed by North et al. [Nor06] called insight-based evaluation, which is also employed in our user study. Here, users verbally comment on everything they see or experience during their exploration with the user interface. This so-called think-aloud protocol serves as basis for the comparative evaluation. Insights will be coded afterwards and quantified to serve as evaluation measure. Recent approaches, e.g., by Gomez et al. [GGZL14] are combining both, task-based and insight-based evaluation methods, to better estimate which of their visualizations fit best for their objectives.

We conducted an insight-based user study to compare BiCFlows to a visualization approach without biclustering. While there is no formal definition of insights, Saraiya et al. [SND05] describes them as "individual observation about the data by the participant" and North et al. [Nor06] listed some key characteristics to gain a better understanding of them:

- **Complex.** Insights are complex, since they involve not only single data values, but large amounts of data.
- **Deep.** Insights are not present from the beginning, but build up over time, bringing up more questions and thus generating depth.
- **Qualitative.** Insights can be subjective and ambiguous and are thereby not exact.
- **Unexpected.** Insights are often unforeseeable.
- **Relevant.** Insights give domain knowledge relevant meaning, because they are embedded in the data and can connect the data to the existing domain.

To measure these insights, North et al. [Nor06] suggest to get rid of predefined benchmark tests, where the users are told which insights to gain. Instead, insights gained by the users on their own will be observed. For this approach, they defined three concepts:

- Open-ended protocol,
- qualitative insight analysis, and
- domain relevance.

The open-ended protocol ensures that the users can take as long as they need to explore the data. During their exploration, they verbally report their findings, which can then be analyzed afterwards using a qualitative insight analysis. Each of these findings is counted as new insight, which will then be coded. Austin and Sutton et al. [AS14] describe coding as a process, where findings of all users are taken to determine similar themes

or ideas. Those can then be used to form thematic categories. Using a coding method to assign each insight to a specific category enables the creation of quantified metrics and further statistical analyses. Among others, the following codes were defined in our user study: entities, transaction sums, or geographical connections. The complete coding scheme will be explained in detail in Section 7.2.

In our user study, we were mainly interested if users can gain more insight using BiCFlows compared to an unclustered approach and whether this increased insight also comes with a higher cognitive effort. We therefore state our two major hypotheses, which we further sub-divided:

H1: *With BiCFlows, users will gain more insights.*

We reason that users are able to explore the data in a more structured way using BiCFlows, because of its clustering approach. Thus, we assume that users will generate more insights. More specifically, we expect that they will find more entities during their exploration. We also believe that users will not only be looking for already known, but will also discover unknown entities or unexpected information. Because of the hierarchical biclustering and the ability to navigate within it, we also expect that there will be more mentions of entities with smaller transaction sums. The biclustered structure could lead to reasoning about commonalities and links between entities. Altogether, we assume that BiCFlows encourages the users to invest more time on their exploration. We therefore expect the following results:

H1.1: With BiCFlows, users will mention more entities and their transaction sums.

H1.2: With BiCFlows, users will mention more entities with smaller transaction sums.

H1.3: With BiCFlows, users will make more links between entities or find more commonalities between them.

H1.4: With BiCFlows, users will discover more unknown entities and unexpected information.

H1.5: With BiCFlows, users will spend more time on their exploration.

H2: *BiCFlows will be perceived as more complex.*

We assume that the biclustered structure will also come with a drawback in user experience. We reason that BiCFlows will not be perceived as intuitive as the alternative user interface and thus will be experienced as more complex.

7.1 Tasks

To analyze advantages and disadvantages of BiCFlows in comparison to already existing approaches, the Cut-Off approach that we introduced in Section 6.1 was used. The users were presented with BiCFlows and the Cut-Off approach to explore data of the Media Transparency Database. Their task was to explore the data with each interface in a think-aloud protocol. This means that they should verbally comment on everything they do and observe during their exploration [Nie94]. After they finished exploring the data with an interface, they had to additionally answer a questionnaire regarding the interface's usability. We employed the system usability scale (SUS) [Bro96], a questionnaire containing the following ten statements that can be rated on a five-point Likert-scale ranging from *Strongly Disagree*, *Somewhat Disagree*, *Neutral*, *Somewhat Agree* to *Strongly Agree*:

1. I think I would like to use this tool frequently.
2. I found the tool unnecessarily complex.
3. I thought the tool was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this tool were well integrated.
6. I thought there was too much inconsistency in this tool.
7. I would imagine that most people would learn to use this tool very quickly.
8. I found the tool very cumbersome to use.
9. I felt very confident using the tool.
10. I needed to learn a lot of things before I could get going with this tool.

The score of every statement, in a range from 0 to 4, contributes to the calculation of the final score. For every statement with an odd number, the contribution is the scale position minus 1 and for every statement with an even number, the contribution is 5 minus the scale position. The sum of all ten statement scores multiplied by 2.5 gives the final SUS score. This score ranges from 0 to 100 where a higher score indicates better usability. To get an understanding of what a certain score means in terms of usability, Bangor et al. [BKM09] evaluated SUS by comparing the final score with an adjective scale. They determined the score range for seven subjective labels: *Worst Imaginable* (0-12), *Awful* (13-20), *Poor* (21-36), *OK* (37-51), *Good* (51-70), *Excellent* (71-85) and *Best Imaginable* (86-100). More generally, they determined tools scoring below 50 as unacceptable, between 70-80 as better and above 90 as superior ones [BKM08]. In our evaluation, we compared not only the final scores, but every statement score separately.

7.2 Design

The study was conducted using the Mozilla Firefox web browser on a 27" monitor. In the beginning of the study, users had to fill in a consent form, followed by a demographic questionnaire and then they were asked to read a printed task description. We used a within-subjects design with the two user interfaces BiCFlows (BiC) and the Cut-Off approach (CO) as independent variable. To compensate the learning effect during the study, we counterbalanced the order of the visualizations, as well as the task assignments to the visualizations, as shown in Table 7.1. Since we wanted to use the Media Transparency Database for both interfaces, we varied the data by selecting only a specific legal basis (§2 or §4) for each run. For §2, there are 1226 legal entities and 3544 media organizations, with a modularity of 0.39 for nine clusters. §4 includes 68 legal entities and 885 media organizations, with a modularity of 0.62 for nine clusters. Thus §4 represents a smaller data set with clusters that are more coherent and share less connections among each other.

User	**Run 1**	**Run 2**
1	BiC §2	CO §4
2	CO §2	BiC §4
3	BiC §4	CO §2
4	CO §4	BiC §2

Table 7.1: Counterbalancing table for the first four users.

During the users' exploration, their comments as well as their interactions with the interface were recorded. The recording was stopped after the user thought that there were no more observations to report. Before the users started exploring the Media Transparency Database, they were given a tutorial about the interaction techniques and a test-dataset for each interface that allowed them to familiarize themselves with its functionality. During this test run nothing was recorded and the users could take as much time as they need as well as ask any questions concerning the study or the user interface. Besides their interaction, like list clicks, bar chart clicks, and their overall exploration time, we also logged the users' ratings from the post-study questionnaire.

7.3 Coding of Think-Aloud Transcripts

After transliterating all recordings, we performed open coding on the users' insights. We grouped utterances into ten categories:

- **Entities.** A mentioned legal entity or media organization.
- **Sums.** Mentioned transaction sums between one legal entity and one media organization, or a total sum spent by a legal entity or received by a media organization.
- **Unexpected findings.** Unexpected findings or astonishments, i.e., "I can't believe *Stadt Wien* spends that much money." or "*Heute* receives that much money? - That's madness!".
- **Unknown entities.** Entities that were unknown to the user, i.e., "*a3ECO*? - Never heard of it before." or "What's *KT1*?"
- **Duplicates.** Discovered entities with same or similar name, i.e., *google.at* and *google.de*, where users explicitly mentioned that these are the same.
- **Time.** Quarters, years, or periods mentioned.
- **Comparisons.** Comparisons between entities or time periods, i.e., "*ÖBB* spent € 19 million, but compared to *Stadt Wien* that's nothing." or "*Österreich Werbung* spent € 20 million in total, but in the fourth quarter of 2016 only € 17,500."
- **Reasoning.** Reasonings made on the basis of certain observations, i.e., "*Heute* receives less from *Land Niederösterreich* than from *Stadt Wien*, most likely because *Heute* is only available in Vienna's subways." or "*Heute*, *Krone*, and *Österreich* receive the most money, that's probably because they have the most readers."
- **Geographical connection.** Geographical connections made for certain entities, i.e., "*DORF TV* is probably from Upper Austria too, because it's in the same group as other media organizations from Upper Austria."

We then used the number of insights per category to check our hypotheses with further statistical analysis. We made an additional count of unique entities, thus eliminating all multiple entities mentioned.

7.4 Participants

Twelve users participated in the study (four female, eight male), aged 25 to 56. Only one of the users has a background in computer science, but all of them use computers and the Internet on a daily basis. Eight users stated that they have very few experiences with scientific or information visualization, three apply them sometimes, and one user very often. Furthermore, only one user had prior knowledge of the Media Transparency Database, two have heard of it before, and nine did not know it at all.

7.5 Results

If we have a look at the number of mentioned entities, we notice that both the total number of entities and the number of unique entities mentioned are on average higher for BiC than for CO (see Figure 7.1). A Wilcoxon Signed-Rank test showed that there is no significant difference for the total number of entities ($Z = 22$, $p = .182$), but for the number of unique entities mentioned ($Z = 10.5$, $p = .045$). Since the number of unique entities do not contain multiple mentions, they are more meaningful than the total number of entities, where the same entity could have been mentioned over and over again. For the mentioned transaction sums we also found a significant difference ($Z = 1.5$, $p = .005$). When comparing the number of sums mentioned, we see that there were more mentions using BiC than CO (see Figure 7.1c). *We can thereby confirm our hypothesis H1.1: Users mention more entities and transaction sums using BiCFlows.*

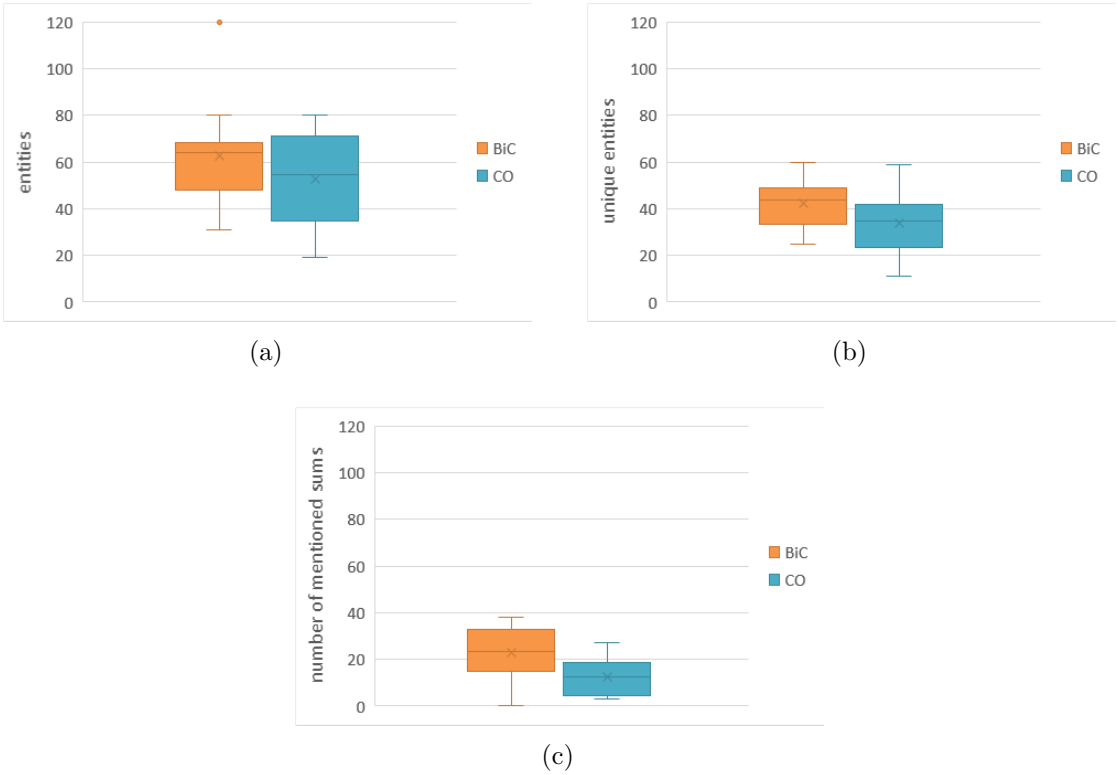


Figure 7.1: Boxplots showing the number of insights per interface for: (a) total number of entities, (b) unique entities and (c) mentioned transaction sums.

To test hypothesis H1.2, we calculated the quartiles of the uniquely mentioned entities. This allowed us to examine if there were more mentions of entities with smaller transaction sums. When comparing the quartiles of uniquely mentioned entities with each other (see Figure 7.2), we see that the means for both $Q1$ are equal (1.17) and for $Q2$, $Q3$, and $Q4$ BiC has only slightly higher averages. Furthermore, a Wilcoxon Signed-Rank test for

each of the quartiles showed no significant difference for any of them: $Q1$ ($Z = 22$, $p = .952$), $Q2$ ($Z = 19$, $p = .383$), $Q3$ ($Z = 31$, $p = .859$) and $Q4$ ($Z = 9.5$, $p = .066$). *This disproves our hypothesis H1.2: Users do not mention more entities with smaller transaction sums using BiCFlows.*

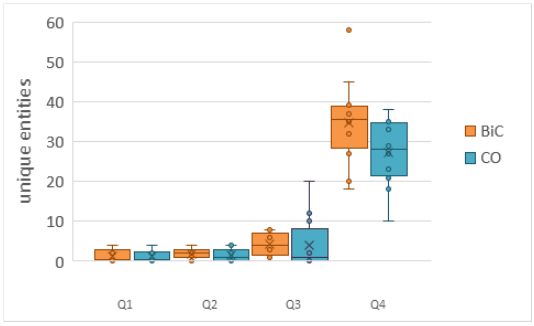


Figure 7.2: Boxplot showing the number of unique entities split into quartiles.

To test hypothesis H1.3, we compared the mentions of geographical connections, since they indicate that the users found links through the user interface that are not clearly visible in the data. An example is the following comment from a user *"So, in this group there are mostly legal entities from Lower Austria and they understandably advertise mostly in newspapers from Lower Austria."*, where she observed that certain legal entities only advertise in a federal state. Figure 7.3 discloses that users were only able to establish these geographical connections while using BiC. *This confirms our hypothesis H1.3: Users made more links between entities using BiCFlows.*

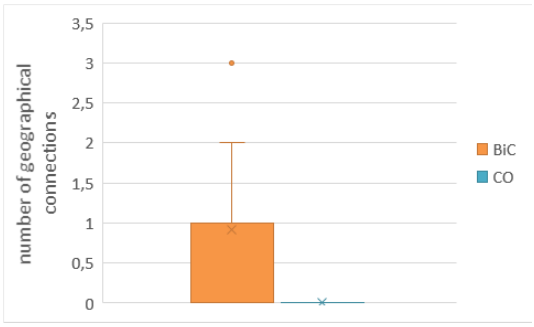
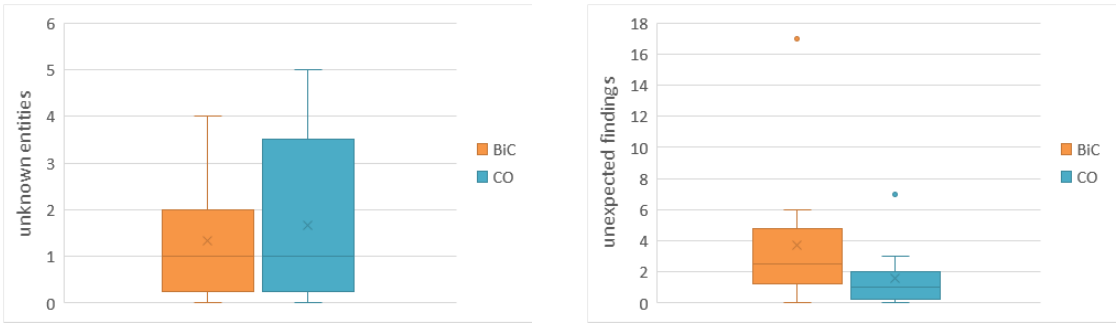


Figure 7.3: Boxplot showing the number of geographical connections made.

For hypothesis H1.4, we compared the number of unknown entities and unexpected findings. Figure 7.4a shows that slightly more unknown entities were mentioned using CO, but a Wilcoxon Signed-Rank test showed that there is no significant difference ($Z = 18.5$, $p = .633$). Looking at the unexpected findings, which include astonished and disbelieving reactions during exploration, we see that there were more findings on average with BiCFlows than with CO (see Figure 7.4b). A Wilcoxon Signed-Rank test also showed a significant difference ($Z = 8$, $p = .045$). *Thus, we can partially confirm our*

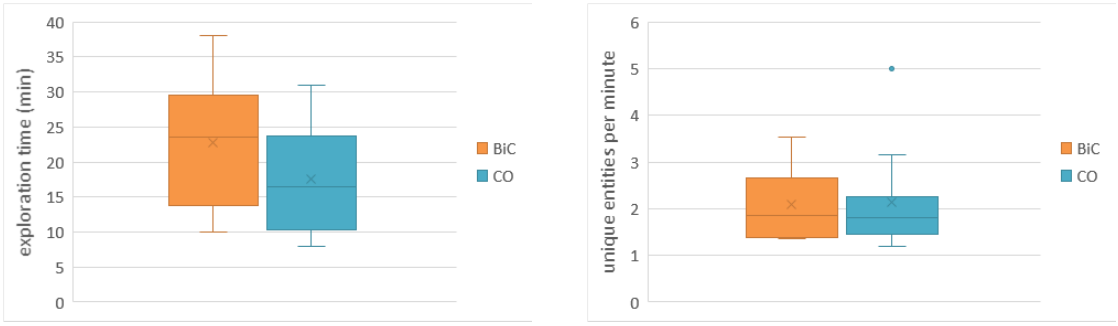
hypothesis H1.4: Users discovered more unexpected information using BiCFlows, but did not find more unknown entities.



(a) (b)

Figure 7.4: Boxplots showing the number of insights per interface for: (a) number of unknown entities, (b) unexpected findings.

To test hypothesis H1.5, we compared the time each user spent exploring the two different interfaces. Figure 7.5a shows that users spent more time on average exploring the data using BiC (23 min) than CO (17.5 min). A Wilcoxon Signed-Rank test also showed a significant difference ($Z = 6.5$, $p = .032$). When comparing the number of unique entities mentioned per minute (see Figure 7.5b), we see that on average CO (2.13) has more than BiC (2.08) and when comparing the median BiC (1.86) has more then CO (1.81). However, they differ only marginally and a Wilcoxon Signed-Rank test also showed no significant difference ($Z = 35$, $p = .754$). *This confirms our hypothesis H1.5: Users invested more time in exploring the data using BiCFlows.*



(a) (b)

Figure 7.5: Boxplot showing: (a) the minutes users spent exploring and (b) the number of unique entities mentioned per minute.

For the sake of completeness, we also tested the remaining categories: duplicates, temporal mentions, comparison, and reasoning. Figure 7.6 reveals that BiC generated more insights on average than CO in all remaining categories, but their differences are not significant:

duplicates ($Z = 2$, $p = .257$), temporal mentions ($Z = 14$, $p = .310$), comparison ($Z = 16$, $p = .774$) and reasoning ($Z = 20$, $p = .234$).

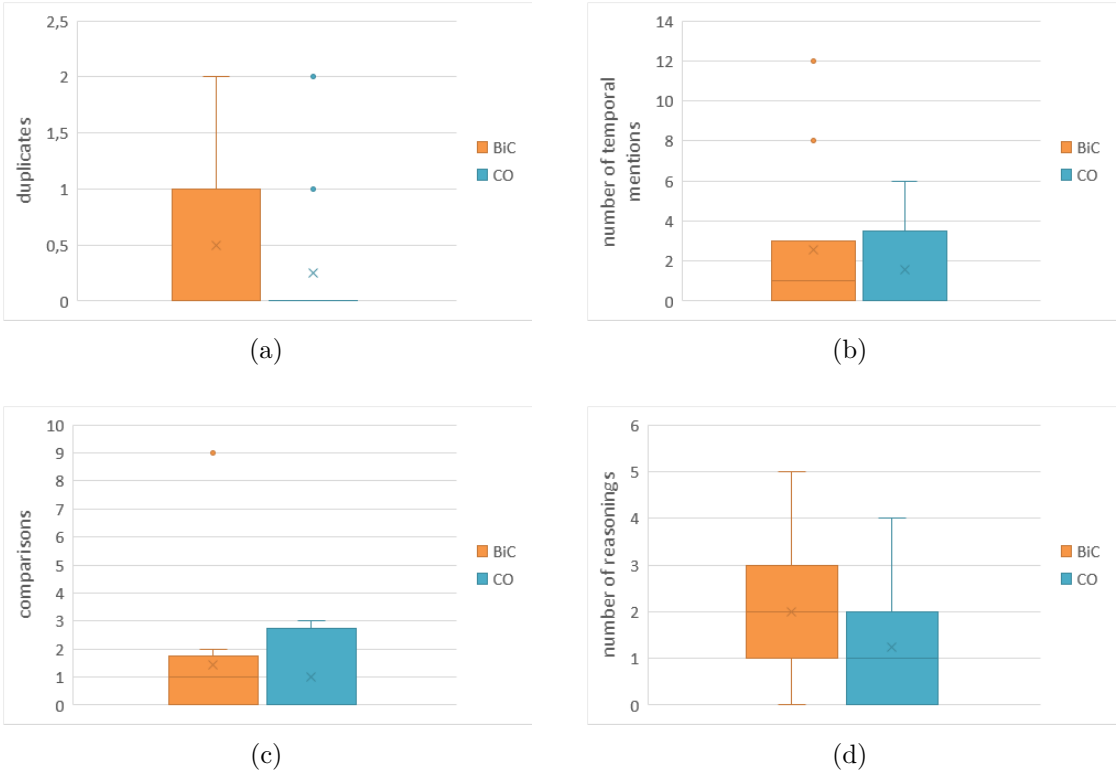


Figure 7.6: Boxplots showing the number of insights per interface for: (a) duplicates, (b) number of temporal mentions, (c) comparisons, (d) reasonings.

Finally, we compared the users' ratings of the SUS questionnaire, to test hypothesis H2. A Wilcoxon Signed-Rank test showed a significant difference for the final SUS score ($Z = 4$, $p = .028$), where CO (82) scores higher than BiC (72) (see Figure 7.7a). After testing every statement of the SUS questionnaire separately, only three of them showed a significant difference, namely *"I found the tool unnecessarily complex"* ($Z = 0$, $p = .008$), *"I thought the tool was easy to use"* ($Z = 0$, $p = .038$) and *"I felt very confident using the tool"* ($Z = 0$, $p = .014$). Comparing the averages of these three statements, respectively (see Figure 7.7), showed that the user felt that BiC (2.0) was more complex than CO (1.5), CO (4.42) was easier to use than BiC (3.83) and they felt more confident using CO (4.25) than BiC (3.5). *This confirms our hypothesis H2: Users perceived BiCFlows as more complex than the Cut-Off approach.*

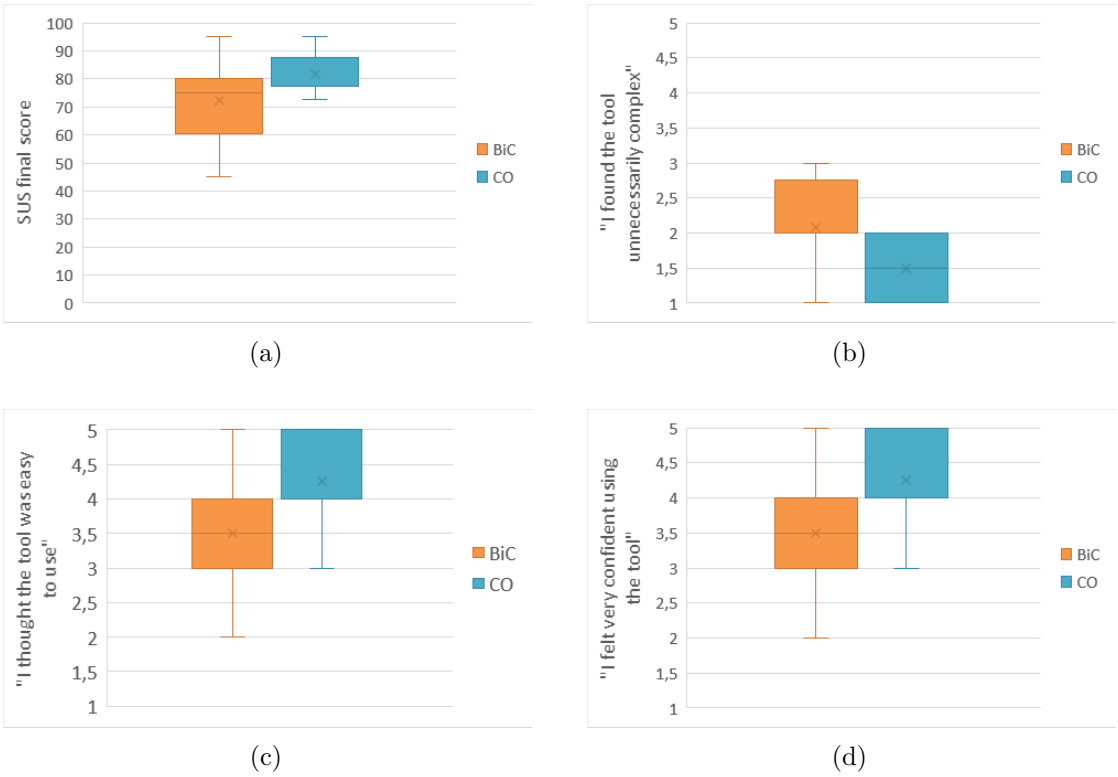


Figure 7.7: Boxplots showing the ratings of the SUS questionnaire: (a) final score, (b) "I found the tool unnecessarily complex", (c) "I thought the tool was easy to use", (d) "I felt very confident using the tool".

7.6 Discussion

Our study showed that we could confirm most of our hypotheses. In the following, we will discuss some of our results.

If we have a look at the comparison between the mentioned transaction sums that we examined in H1.1, we see that the users mentioned more sums in BiCFlows. We reason that this is mainly because it correlates with the mentioned entities, i.e., if they mention more entities, they will also mention their corresponding sums. The fact that the number of mentioned entities and the number of mentioned sums are not equal, could be due to the fact that users are not always interested in the sum, but rather the entity itself.

In H1.2 we expected the users to find more entities with smaller transaction sums using BiCFlows, but our user study disproved this. We assumed that there would be fewer findings with CO due to the fact that entities with small sums always get aggregated and thus cannot be explored, except if users are explicitly looking for these entities in the tables. BiCFlows offers the possibility to explore these entities, but to find them users have to go multiple levels down in the cluster hierarchy to reach a level where entities

with small transaction sums are present. However, many users only went down one or two levels and thus never advanced far enough to explore these entities. That is the reason why the number of entities mentioned using BiCFlows is as low as using CO for $Q1$ and $Q2$.

If we have a look at H1.3, where we compared unexpected findings, we saw that more unexpected information was discovered using BiCFlows. However, what did the users find and why were there more findings in BiCFlows? When looking at the raw data, we observe similar findings for BiCFlows and CO. During every session, users were very surprised when looking at advertising expenditures and seeing *Kronen Zeitung*, *Heute*, and *Österreich* at the very top of the receiving media organizations. Furthermore, they were irritated about the fact that *ORF1* receives less money than *ORF2*, since they assumed *ORF1* will be watched by more people and is thus more interesting for advertisers than *ORF2*. Another observation that astonished many users is the fact that the *Bundeskanzleramt* is funding many Croatian clubs. However, since it is responsible for the funding of ethnic groups and Burgenland-Croatians are the second largest minority in Austria, it is not that surprising. Regarding the higher number of findings using BiCFlows, the following could be observed. Although *Stadt Wien* was mentioned in both, BiCFlows and CO, as the legal entity having the largest advertising budget, it was mentioned more often using BiCFlows in matters of having the most receivers. We assume that is because in BiCFlows, *Stadt Wien*'s connections are more present than in CO. Figure 7.8 shows how *Stadt Wien* will be perceived in BiCFlows, not only showing up as the entity with the largest sum, but also revealing its huge number of receivers in comparison to CO (see Figure 7.9).

As tested in H1.5, users spent more time exploring the data on average with BiCFlows than with CO. This means that when using CO, users had the feeling that they will not gain more insight if they keep exploring and thus ended the session, whereas BiCFlows encouraged them to keep exploring. This can also explain the higher number of entities mentioned in H1.1. The rate of mentioned unique entities per minute is approximately the same, which emphasizes the assumption that the users found more entities because they also explored longer.

When investigating the differences in complexity of the two user interfaces (H2), we found out that users perceived BiCFlows as more complex. We assumed that the hierarchical clustering and its interaction possibilities would be harder to understand. In fact, some users stated afterwards that in the beginning of the session they found the clustering concept irritating, but gained an understanding of it during their exploration.

As for the SUS final score, BiCFlows scored 72 and CO scored 82. Although BiCFlows has a lower average rating than CO, it is still in a good position. The subjective rating defined by Bangor et al. [BKM09] and described in Section 7.1, characterizes both user interfaces as *Excellent* (71-85), even though their scores are on the lower and upper ends of the range, respectively.

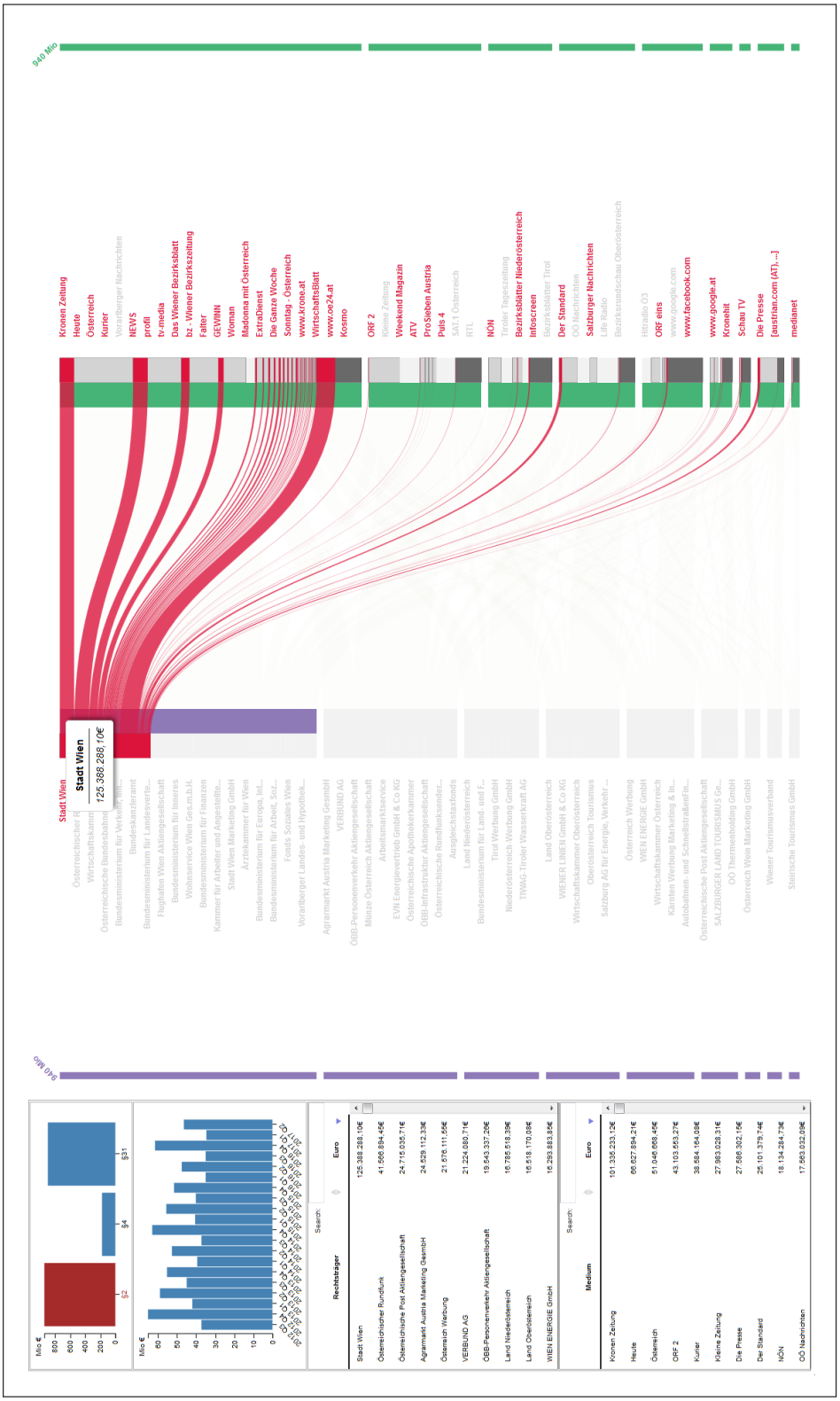


Figure 7.8: BiCFlows with selected entity *Stadt Wien*.

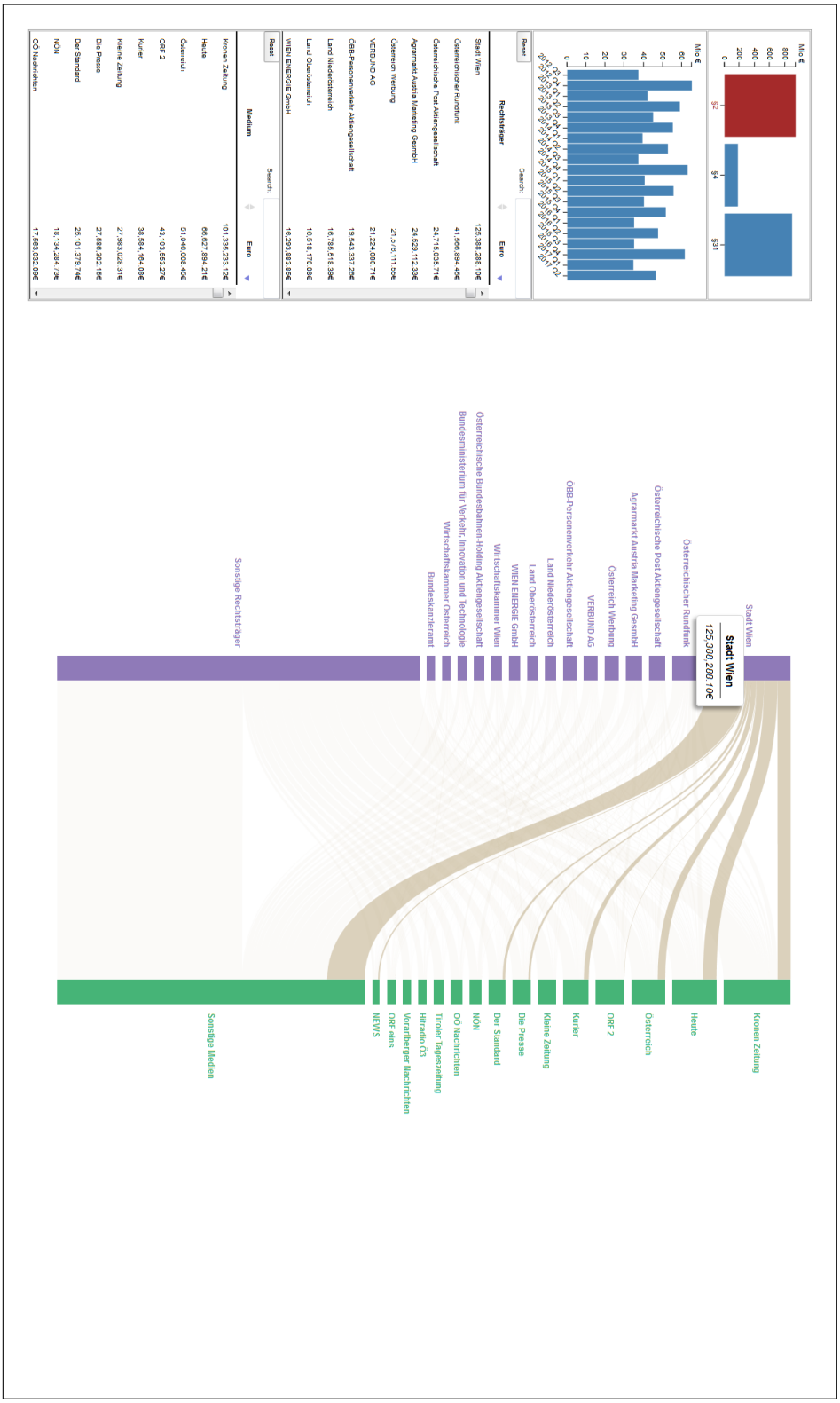


Figure 7.9: Cut-Off approach with selected entity *Stadt Wien*.

7.7 Limitations

One limitation of our user study is that there is a potential confounding factor regarding the entity labels. While in the Cut-Off approach every entity has a label (see Figure 6.1), we chose a different approach in BiCFlows as described in Section 4.3. This results in approximately three times more labels in BiCFlows than in the Cut-Off approach. In reference to our results confirming H1.2, we therefore cannot exclude the possibility of users finding more entities due to the fact that there were more labels visible initially.

Furthermore, we did not evaluate if the clustering concept was overestimated in the sense that user drew wrong conclusions with it. An example would be that users see a cluster including many newspapers from the same region and therefore conclude that this specific cluster represents only this regions newspapers and no others.

CHAPTER 8

Conclusion

In this thesis, we developed BiCFlows, a novel exploration interface for large bipartite graphs. We presented the limitations and difficulties of existing approaches and showed how BiCFlows can overcome them. We discussed different use cases such as the use for exploring publication or movie data. The main motivation however, was the Media Transparency Database that we also used as data source for our evaluation. In our user study, we could confirm our hypothesis that the employed clustering approach helped users to gain more insight than with unclustered approaches. With BiCFlows, they mentioned more entities and their corresponding sums. Additionally, the exploratory approach helped them to discover unexpected findings. Moreover, we could confirm that BiCFlows can help to derive relations between entities that are not present in the raw data, like geographical connections. However, the deeper insights gained through the clustered representation also comes with a drawback, as it is perceived as more complex than unclustered visualizations.

Although the usability of BiCFlows was rated lower than the visualization we compared it with, it was still perceived as a very useful tool by the study participants, who mainly have no background in computer science. This gives us confidence that BiCFlows is also useful to non-specialists without a technical or scientific background.

When comparing the two approaches for different tasks, we would argue that BiCFlows is definitely better in its ability of supporting untargeted exploration and finding similar entities. For quickly determining specific predefined questions, however, the Cut-Off approach will be of better use. Therefore, it would be reasonable for future works to integrate the ability to select multiple entities and directly compare them with each other also into BiCFlows. This would combine the strengths of both approaches and would lead to a more comprehensive tool.

Bibliography

[AA06] Natalia Andrienko and Gennady Andrienko. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer Berlin Heidelberg, 2006.

[AES05] Robert Amar, James Eagan, and John Stasko. Low-Level Components of Analytic Activity in Information Visualization. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pages 111–117, Washington, DC, USA, 2005. IEEE Computer Society.

[AMA08] Daniel Archambault, Tamara Munzner, and David Auber. GrouseFlocks: Steerable Exploration of Graph Hierarchy Space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, July 2008.

[App17] Joe Apple. List of Random Names. `http://listofrandomnames.com`, 2017. [Online; accessed Nov-2017].

[ARN15] Melissa Ailem, François Role, and Mohamed Nadif. Co-clustering Document-term Matrices by Direct Maximization of Graph Modularity. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, CIKM '15, pages 1807–1810, New York, NY, USA, 2015. ACM.

[AS14] Zubin Austin and Jane Sutton. Qualitative Research: Getting Started. *The Canadian Journal of Hospital Pharmacy*, 67(6):436–440, 2014.

[BBD09] Fabian Beck, Michael Burch, and Stephan Diehl. Towards an Aesthetic Dimensions Framework for Dynamic Graph Visualisations. In *2009 13th International Conference Information Visualisation*, pages 592–597, July 2009.

[BBDW17] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A Taxonomy and Survey of Dynamic Graph Visualization. *Computer Graphics Forum*, 36(1):133–159, January 2017.

[BBP+06] Simon Barkow, Stefan Bleuler, Amela Prelić, Philip Zimmermann, and Eckart Zitzler. BicAT: a biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, May 2006.

[Ber06] Pavel Berkhin. A Survey of Clustering Data Mining Techniques. In *Grouping Multidimensional Data*, pages 25–71. Springer, Berlin, Heidelberg, 2006.

[Bic10] Charles-Edmond Bichot. Co-clustering documents and words by minimizing the normalized cut objective function. *Journal of Mathematical Modelling and Algorithms*, 9(2):131–147, 2010.

[Bic17] Biclustering module of Scikit-learn: Machine Learning in Python. `http://scikit-learn.org/stable/modules/biclustering.html`, 2017. [Online; accessed Oct-2017].

[BKH05] Fabian Bendix, Robert Kosara, and Helwig Hauser. Parallel Sets: Visual Analysis of Categorical Data. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 133–140. IEEE, 2005.

[BKM08] Aaron Bangor, Philip Kortum, and James Miller. An Empirical Evaluation of the System Usability Scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[BKM09] Aaron Bangor, Philip Kortum, and James Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Studies*, 4(3):114–123, May 2009.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 - Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, December 2011.

[Bos17] Michael Bostock. D3: Data-Driven Documents. `https://d3js.org`, 2017. [Online; accessed Oct-2017].

[Bro96] John Brooke. SUS - A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[Bun17] Bundeskanzleramt. Medienkooperations- und -förderungs-Transparenzgesetz. `https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20007610`, 2017. [Online; accessed Oct-2017].

[Car08] Sheelagh Carpendale. Evaluating Information Visualizations. In *Information Visualization*, Lecture Notes in Computer Science, pages 19–45. Springer, Berlin, Heidelberg, 2008.

[CC00] Yizong Cheng and George Church. Biclustering of expression data. *Proceedings. International Conference on Intelligent Systems for Molecular Biology*, 8:93–103, 2000.

[CGSQ11] Nan Cao, David Gotz, Jimeng Sun, and Huamin Qu. DICON: Interactive Visual Analysis of Multidimensional Clusters. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2581–2590, December 2011.

[CSBT09] Ya-Xi Chen, Rodrigo Santamaría, Andreas Butz, and Roberto Therón. Tagclusters: Semantic aggregation of collaborative tags beyond tagclouds. In *International Symposium on Smart Graphics*, pages 56–67. Springer, 2009.

[CT05] Kristin A. Cook and James J. Thomas. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005.

[CT06] Kristin A. Cook and James J. Thomas. A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13, January 2006.

[CY00] Chaomei Chen and Yue Yu. Empirical studies of information visualization: a meta-analysis. *International Journal of Human-Computer Studies*, 53(5):851–866, November 2000.

[CZWL08] Weiwei Cui, Hong Zhou, Pak Chung Wong, and Xiaoming Li. Geometry-Based Edge Clustering for Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):8, 2008.

[der18] derStandard.at. Medientransparenz: Regierungswerbung. `https://derstandard.at/r1291455117217/Medientransparenz-Regierungswerbung`, 2018. [Online; accessed Mar-2018].

[Dhi01] Inderjit S. Dhillon. Co-clustering Documents and Words Using Bipartite Spectral Graph Partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 269–274, New York, NY, USA, 2001. ACM.

[DMRW11] Maxime Dumas, Michael J. McGuffin, Jean-Marc Robert, and Marie-Claire Willig. Optimizing a Radial Layout of Bipartite Graphs for a Tool Visualizing Security Alerts. In *Graph Drawing*, pages 203–214. Springer, Berlin, Heidelberg, September 2011.

[DRRD12] Marian Dörk, Nathalie Henry Riche, Gonzalo Ramos, and Susan Dumais. PivotPaths: Strolling through Faceted Information Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2709–2718, December 2012.

[EB15] Thomas E. Bartlett. Co-modularity and Co-community Detection in Large Networks. *arXiv preprint arXiv:1511.05611*, November 2015.

[EF10] Niklas Elmqvist and Jean-Daniel Fekete. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, May 2010.

[FFHW15] Florian Fittkau, Santje Finke, Wilhelm Hasselbring, and Jan Waller. Comparing Trace Visualizations for Program Comprehension through Controlled Experiments. In *2015 IEEE 23rd International Conference on Program Comprehension*, pages 266–276, May 2015.

[FGK12] Darya Filippova, Aashish Gadani, and Carl Kingsford. Coral: an integrated suite of visualizations for comparing clusterings. *BMC Bioinformatics*, 13:276, October 2012.

[Fou18] Python Software Foundation. Python programming language. `https://www.python.org`, 2018. [Online; accessed Feb-2018].

[FSB+13] Patrick Fiaux, Maoyuan Sun, Lauren Bradel, Chris North, Naren Ramakrishnan, and Alex Endert. Bixplorer: Visual Analytics with Biclusters. *Computer*, 46(8):90–94, August 2013.

[FSM17] Role François, Morbieu Stanislas, and Nadif Mohamed. Coclust: a Python package for co-clustering. `https://coclust.readthedocs.io`, 2017. [Online; accessed Oct-2017].

[GFC04] Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. In *IEEE Symposium on Information Visualization*, pages 17–24, 2004.

[GGZL14] Steven R. Gomez, Hua Guo, Caroline Ziemkiewicz, and David H. Laidlaw. An insight- and task-based methodology for evaluating spatiotemporal visual analytics. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 63–72, October 2014.

[GKL+13] Sohaib Ghani, Bum Chul Kwon, Seungyoon Lee, Ji Soo Yi, and Niklas Elmqvist. Visual Analytics for Multimodal Social Network Analysis: A Design Study with Social Scientists. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2032–2041, December 2013.

[GL08] Jiajun Gu and Jun S. Liu. Bayesian biclustering of gene expression data. *BMC genomics*, 9 Suppl 1:S4, 2008.

[GYZ13] Jonathan L. Gross, Jay Yellen, and Ping Zhang. *Handbook of Graph Theory, Second Edition.* Chapman & Hall/CRC, 2nd edition, 2013.

[Har72] John A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

[HB03] Mark Harrower and Cynthia A. Brewer. ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps. *The Cartographic Journal*, 40(1):27–37, June 2003.

[HFM07] Nathalie Henry, Jean-Daniel Fekete, and Michael J. McGuffin. NodeTrix: a Hybrid Visualization of Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, November 2007.

[HMM00] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January 2000.

[HP96] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84. ACM, 1996.

[HSS17] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. NetworX: Software for complex networks. `https://networkx.github.io`, 2017. [Online; accessed Oct-2017].

[HvW09] Danny Holten and Jarke J. van Wijk. Force-Directed Edge Bundling for Graph Visualization. *Computer Graphics Forum*, 28(3):983–990, June 2009.

[HW12] Steve Haroz and David Whitney. How Capacity Limits of Attention Influence Information Visualization Effectiveness. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2402–2410, December 2012.

[IEE18] IEEE. IEEE Taxonomy 2018. `https://www.ieee.org/documents/taxonomy_v101.pdf`, 2018. [Online; accessed Feb-2018].

[IHK+17] Petra Isenberg, Florian Heimerl, Steffen Koch, Tobias Isenberg, Panpan Xu, Charles D. Stolper, Michael M. Sedlmair, Jian Chen, Torsten Möller, and John Stasko. vispubdata.org: A Metadata Collection about IEEE Visualization (VIS) Publications. *IEEE transactions on visualization and computer graphics*, 23(9):2199–2206, 2017.

[IHK+18] Petra Isenberg, Florian Heimerl, Steffen Koch, Tobias Isenberg, Panpan Xu, Charles D. Stolper, Michael M. Sedlmair, Jian Chen, Torsten Möller, and John Stasko. Visualization Publication Data Collection. `http://www.vispubdata.org`, 2018. [Online; accessed Feb-2018].

[IIS+18] Petra Isenberg, Tobias Isenberg, Michael Sedlmair, Jian Chen, and Torsten Möller. KeyVis: Search for VIS paper keywords. `http://keyvis.org`, 2018. [Online; accessed Feb-2018].

[Ins09] Alfred Inselberg. Parallel Coordinates. In *Encyclopedia of Database Systems*, pages 2018–2024. Springer, Boston, MA, 2009.

[Jai10] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, June 2010.

[jF17] The jQuery Foundation. jQuery. `https://jquery.com`, 2017. [Online; accessed Oct-2017].

[JTZ04] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, November 2004.

[KBCG03] Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. Spectral Biclustering of Microarray Cancer Data: Co-clustering Genes and Conditions. *Genome Research*, 13:703–716, 2003.

[KM13] Robert Kosara and Jock Mackinlay. Storytelling: The Next Step for Visualization. *Computer*, 46(5):44–50, May 2013.

[LF06] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.

[LN11] Lazhar Labiod and Mohamed Nadif. Co-clustering for Binary and Categorical Data with Maximum Modularity. In *2011 IEEE 11th International Conference on Data Mining*, pages 1140–1145, December 2011.

[Mag17] Paroli Magazin. `http://www.paroli-magazin.at/105`, 2017. [Online; accessed Oct-2017].

[Mir98] Boris Mirkin. Mathematical classification and clustering: From how to what and why. In *Classification, Data Analysis, and Data Highways*, pages 172–181. Springer, 1998.

[Mis06] Kazuo Misue. Drawing Bipartite Graphs As Anchored Maps. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60*, APVis '06, pages 169–177, Darlinghurst, Australia, 2006. Australian Computer Society, Inc.

[MO04] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, January 2004.

[New06] Mark E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

[NG04] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, 69(2):026113, February 2004.

[Nie94] Jakob Nielsen. *Usability Engineering*. Interactive Technologies. Elsevier Science, 1994.

[Nor06] Chris North. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, 26(3):6–9, May 2006.

[OKSK16] Yosuke Onoue, Nobuyuki Kukimoto, Naohisa Sakamoto, and Koji Koyamada. Minimizing the Number of Edges via Edge Concentration in Dense Layered Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1652–1661, June 2016.

[Oli17] Travis E. Oliphant. NumPy: a fundamental package for scientific computing with Python. `http://www.numpy.org`, 2017. [Online; accessed Oct-2017].

[Pfa17] David Pfahler. Investigating Media Transparency. `https://www.pfahler.at/mtdb2`, 2017. [Online; accessed Oct-2017].

[PGAR15] Beatriz Pontes, Raúl Giráldez, and Jesús S. Aguilar-Ruiz. Biclustering on expression data: A review. *Journal of Biomedical Informatics*, 57:163–180, October 2015.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[PXYH05] Doantam Phan, Ling Xiao, Ron Yeh, and Pat Hanrahan. Flow map layout. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 219–224, October 2005.

[Rah17] Md. Saidur Rahman. *Basic Graph Theory*. Undergraduate Topics in Computer Science. Springer International Publishing, Cham, 2017.

[RD10] Nathalie Henry Riche and Tim Dwyer. Untangling Euler Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1090–1099, November 2010.

[RDF06] Manjeet Rege, Ming Dong, and Farshad Fotouhi. Co-clustering Documents and Words Using Bipartite Isoperimetric Graph Partitioning. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 532–541, December 2006.

[Rec17] Rechnungshof. Medientransparenzgesetz. `http://www.rechnungshof.gv.at/sonderaufgaben/medientransparenzgesetz.html`, 2017. [Online; accessed Oct-2017].

[RHF05] Patrick Riehmann, Manfred Hanfler, and Bernd Froehlich. Interactive Sankey diagrams. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 233–240, October 2005.

[RMF12] Sébastien Rufiange, Michael J. McGuffin, and Christopher P. Fuhrman. TreeMatrix: A Hybrid Visualization of Compound Graphs. *Computer Graphics Forum*, 31(1):89–101, February 2012.

[Ron18] Armin Ronacher. Flask: a micro web-framework for Python. `http://flask.pocoo.org`, 2018. [Online; accessed Feb-2018].

[RPNA16] Alexander Rind, David Pfahler, Christina Niederer, and Wolfgang Aigner. Exploring media transparency with multiple views. In *Proceedings of the 9th Forum Media Technology 2016*, pages 65–73. CEUR-WS, 11 2016.

[Sal17] Peter Salhofer. MEHR! Medientransparenz. `https://www.medien-transparenz.at`, 2017. [Online; accessed Oct-2017].

[SGG+14] Marc Streit, Samuel Gratzl, Michael Gillhofer, Andreas Mayr, Andreas Mitterecker, and Sepp Hochreiter. Furby: fuzzy force-directed bicluster visualization. *BMC Bioinformatics*, 15(6):S4, May 2014.

[SGL07] John Stasko, Carsten Görg, and Zhicheng Liu. Jigsaw: Supporting Investigative Analysis through Interactive Visualization. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 131–138, October 2007.

[Shn03] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The Craft of Information Visualization*, pages 364–371. Elsevier, 2003.

[SJUS08] Hans-Jörg Schulz, Mathias John, Andrea Unger, and Heidrun Schumann. Visual Analysis of Bipartite Biological Networks. In *Proceedings of the First Eurographics Conference on Visual Computing for Biomedicine*, EG VCBM'08, pages 135–142, Aire-la-Ville, Switzerland, 2008. Eurographics Association.

[SMNR16] Maoyuan Sun, Peng Mi, Chris North, and Naren Ramakrishnan. BiSet: Semantic Edge Bundling with Biclusters for Sensemaking. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):310–319, January 2016.

[SND05] Purvi Saraiya, Chris North, and Karen Duca. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):443–456, July 2005.

[SNR14] Maoyuan Sun, Chris North, and Naren Ramakrishnan. A Five-Level Design Framework for Bicluster Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1713–1722, December 2014.

[Spr17] SpryMedia. DataTables - a table plug-in for jQuery. https://datatables.net, 2017. [Online; accessed Oct-2017].

[Squ17] Square. Crossfilter. https://square.github.io/crossfilter, 2017. [Online; accessed Oct-2017].

[STQ08] Rodrigo Santamaría, Roberto Therón, and Luis Quintales. BicOverlapper: A tool for bicluster visualization. *Bioinformatics*, 24(9):1212–1213, May 2008.

[TSS02] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(1):136–144, July 2002.

[uTRG17] Rundfunk und Telekom Regulierungs-GmbH. Bekanntgegebene Daten. https://www.rtr.at/de/m/veroeffentl_medkftg_daten, 2017. [Online; accessed Oct-2017].

[Viz17] Viz - A Collection of visualization layouts. https://github.com/NPashaP/Viz, 2017. [Online; accessed Oct-2017].

[vLKS+11] Tatiana von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J. van Wijk, Jean-Daniel Fekete, and Dieter W. Fellner. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum*, 30(6):1719–1749, September 2011.

[WCV11] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, 13(2):22–30, March 2011.

[Wic17] Hadley Wickham. ggplot2movies. https://github.com/hadley/ggplot2movies/blob/master/data-raw/movies.csv, 2017. [Online; accessed Nov-2017].

[WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[WS05] Scott White and Padhraic Smyth. A Spectral Clustering Approach To Finding Communities in Graphs. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 274–285. Society for Industrial and Applied Mathematics, April 2005.

[XCQS16] Panpan Xu, Nan Cao, Huamin Qu, and John Stasko. Interactive visual co-cluster analysis of bipartite graphs. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 32–39, April 2016.

[YCN04] Kevin Y. Yip, David W. Cheung, and Michael K. Ng. HARP: a practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(11), November 2004.

[ZSCC18] Jian Zhao, Maoyuan Sun, Francine Chen, and Patrick Chiu. BiDots: Visual Exploration of Weighted Biclusters. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):195–204, January 2018.