Linnéuniversitetet
Kalmar Växjö

Master Thesis Project

Interactive Visual Analysis of Hypergraphs

Author: Ningrui Chen
Supervisor: Dr. Kostiantyn Kucher
Co-Supervisor: Prof. Dr. Andreas Kerren
Examiner: Dr. Morgan Ericsson
Semester: VT 2021
Course Code: 5DV50E
Subject: Computer Science

Abstract

Access to and understanding of data plays an essential role in the increasingly digital world. Representation and analysis of relations between various data entities, i.e., graph and network structures in the data, is an important problem for various industries. In contrast to simple graphs that focus on edges with two endpoints only, a hypergraph provides a natural method to represent multi-way interactions with an arbitrary number of endpoints for each edge, and it can be a better alternative than a bipartite graph for comparable applications. However, traditional approaches for visually representing hypergraphs are purely static diagrams without support for interaction, which can be complex to perceive and do not scale well with regard to the number of nodes and edges. They are not adequate for representation and interactive exploration of large or dense hypergraph data sets found in real-world applications. The ISOVIS (Information and Software Visualisation) research group at Linnaeus University has previously introduced a novel radial visualization approach for undirected hypergraphs called Onion. The Onion tool focuses on solving the issues of edge clutter, overlaps, and edge crossings. However, certain open challenges and suggestions for improvements were identified for the respective implementation, and there is an opportunity to fill a gap in the hypergraph visualization research by building upon the original Onion approach study. In this thesis project, we implement the new version of the Onion approach based on the principles and challenges established previously. The contributions of this work include evidence regarding the effectiveness and efficiency of a hypergraph comparison technique, the usability of edge bundling in the context of hypergraph exploration tasks, and the scalability of the interactive visualization through an entirely new web-based version of the Onion approach. To obtain the respective results, the new implementation is applied for two case studies involving real-world data sets, and further validated through a user study with several participants. The results of this work can be helpful for researchers of network visualization and practitioners in need of approaches for representing and exploring data that can be modelled as hypergraphs.

Keywords: hypergraph, hyperedge, cardinality of hyperedges, edge bundling, edge crossing, network visualization, information visualization, interaction

Contents

1 Introduction

Access to and understanding of data plays an essential role in the increasingly digital world. Analysis and explanation of relations between various entities and data points is a fundamental problem for various industries [1]. Such relational data can typically be modelled as *graphs* or *networks* (i.e., graphs with additional attributes attached to its constituent *vertices/nodes* or *edges*). People need an intelligible and concise way to illustrate and interact with complex network structures.

Information visualization is a research field that studies visual representations used for data interpretation and effective information communication, such as various charts, and interaction techniques can be used alongside such representations [2]. Visualization is not meant to only provide aesthetically pleasing renderings of the data, but rather, it should help us establish relationships and patterns in our minds that cannot be observed naturally by looking at numerical data directly [3]. Information visualization can be further combined in workflows involving intense application of computational approaches, which is the focus of the visual analytics field [4, 5].

Visual representation of graphs has long been in the focus of the field of graph drawing [6], which is mainly concerned with computation of layouts of so-called node-link diagrams. Within information visualization research, the respective subfield focusing on graphs and networks is known as network visualization, and it is mainly concerned with various representations, interaction techniques, and applications involving such graph and network data, for example, visual analysis of large graphs [7] or social networks [8].

While the simple, ordinary graph model only includes nodes and edges that connect pairs of nodes, there is sometimes a need to represent a relationship between more than two nodes, for example, to indicate the membership of several social network actors in communities [1]. One way of addressing this task with edges connecting two vertices/nodes only is by using a *bipartite graph* [9] that introduces additional nodes corresponding to such communities. The disadvantages of such an approach are the growth of the number of edges and, consequently, more cluttered and space-demanding visual representation, which can make it more difficult for the users to understand the overall structure and particular relations in the graph.

To describe a complicated network for such a scenario, a *hypergraph* is a better option than the bipartite graph. A hypergraph provides a natural method to represent multi-way interactions with an arbitrary number of nodes, i.e., its *hyperedges* can be connected to more than two endpoints at a time [10]. A hypergraph is computationally more efficient and can better model complex non-pairwise relationships for many data sets than simple, ordinary graphs [9]. The hypergraph approach has been demonstrated to be widely applicable in many areas, for example, for efficient storage of large databases on disks and data mining [11].

However, traditional approaches for visual representation of hypergraphs are purely static diagrams without any support for interaction, and they can be complex and do not scale well for larger data sets [11]. They are often not adequate for handling the data sets used in modern real-world applications. The ISOVIS (Information and Software Visualisation) research group at Linnaeus University focuses on the big data challenge by combining human-centered data analysis and interactive visualization. The ISOVIS group has previously introduced the Onion tool [12] that provides a new radial visualization method that can display undirected hypergraphs without clutter, including a set of possibilities to interact with them. As this previous study shows, the Onion approach has advantages over many other hypergraph layout approaches, especially on the performance of hyperedge representation [12]. However, the authors of the study concluded that further work was needed to offer an efficient Onion tool for the visual analysis of hypergraphs, based on certain open challenges and opportunities identified as part of their study.

In this project, we carry out a further study of hypergraph visualization based on the Onion tool. We find evidence about the effectiveness and efficiency of a hypergraph comparison technique, the usability of edge bundling in the context of hypergraph exploration tasks, and the scalability of the interactive visualization through an entirely new web-based version of the Onion approach using JavaScript.

1.1 Background

This section will prove the background knowledge regarding our Onion visualization approach in detail, such as the critical theories and the terms we will use in this report.

1.1.1 Hypergraphs

A hypergraph preserves a network structure in which more than two relationships are involved. In mathematics, the definition of a hypergraph is a graph in which an edge can join any number of vertices. Such edges are thus called hyperedges. Moreover, they can be treated as non-empty subsets of the vertices in the hypergraph. More formally, a hypergraph is a pair $H = (V, A)$, where V is a finite vertex set, and A represents a set of edges that are connected with at least one node [13]. The definition of a hypergraph is related to the definition of a graph $G = (V, E)$ such that each hypergraph H induces a connected subgraph of G. Hypergraph is a model of multi-adic [14] relationships in structures where the traditional pairwise relationship of graphs is insufficient [15]. It has been used in many fields and domains, such as social networks [16], chemical reactions [17], genome [18], and VLSI design [19].

1.1.2 Edge Bundling

Based on the previous study [12], one of the problems of the original interactive Onion tool is that support for edge bundling for ordinary edges (i.e., edges connecting two vertices) is not complete. In this project, we try to find evidence about the usability of edge bundling in the context of hypergraph exploration that suits the Onion tool.

Limited space is one of the reasons that may lead to edge clutter. Edges play an essential role by encoding the relationships of the nodes as well as the associated data. However, graph/network visualizations face edge clutter problems when the number of data items increases. Moreover, edge clutter constitutes a challenge for users with regard to obtaining information and insights from the visualization. Edge bundling methods are one potential solution to this issue.

Edge bundling provides an abstract and uncluttered view of the original edge-cluttered visualization. Different edge bundling algorithms exist, such as geometric graphs, trees, or parallel coordinates plots, where each node has a predefined location that is used as input for generating the curved representations of each edge [20].

1.1.3 Visualization Scalability Issues

In the field of information visualization, scalability is an open problem with multiple facets, as the data sets used in critical real-world applications can potentially include massive networks, huge multi-dimensional heterogeneous data sets, and complex data streams. Representing such data sets can require advanced and rather complicated visualization techniques. However, this issue also leads to the question of perceptual scalability of visualizations [21, 22]. Regarding the limitations of human vision, a human perceives a restricted number of pixels in a specific observed distance. In general, information visualization researchers have limited screen space to use for representing complex data, and it is not always clear if the chosen representation can scale up to larger data sets without becoming difficult to perceive, and also too slow to compute, render, and interact with. In this project, we find evidence about the scalability of the interactive visualization approach implemented in Onion.

1.1.4 The Original Onion Tool

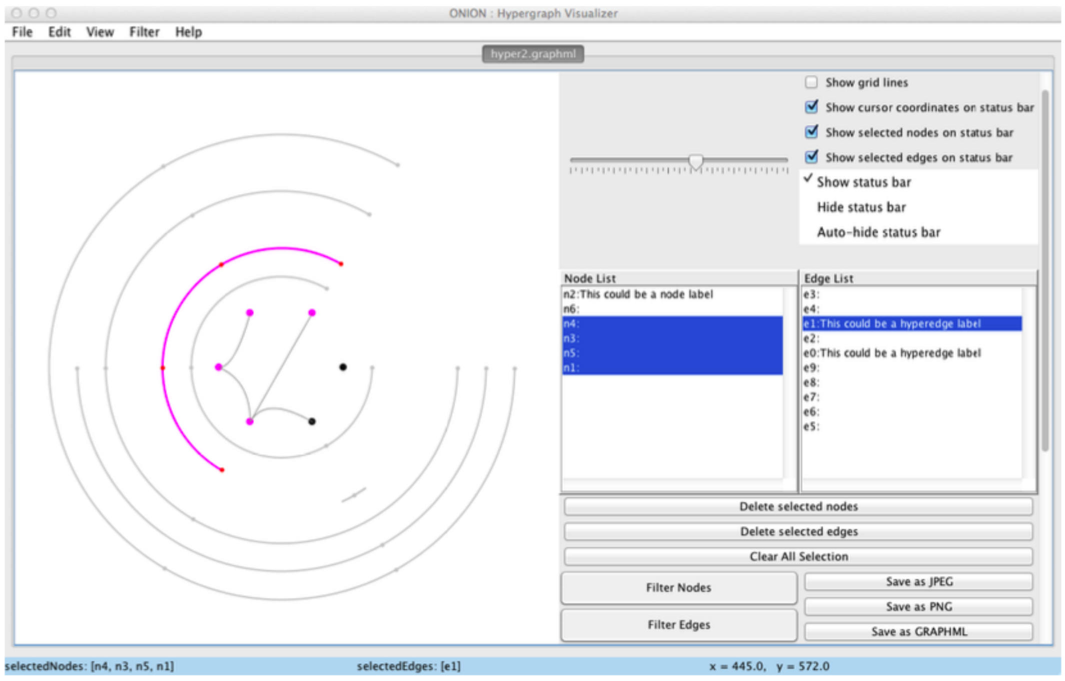
As shown in Figure 1.1, the Onion tool [12] is a hypergraph visualization tool that has been developed by Prof. Dr. Andreas Kerren and Dr. Ilir Jusufi in 2013. The Onion tool focuses on solving the issues of edge clutter, overlaps, and edge crossings for hypergraph data representation. The following requirements guided the prototypical implementation of the original Onion tool [12]:

- User-friendly metaphor that is intuitively understandable;
- GraphML [23] files should specify input hypergraph;
- Support for standard interactions [2, 24], such as zooming, filtering, or reordering of hypergraph elements;
- Hypergraph edges should not overlap (no crossings of hyperedges);
- Hypergraph and ordinary edges (hypergraph of cardinality two) should be treated and shown separately.

Besides, the Onion tool considered several tasks that the visualization and associated interaction techniques should support [12]:

- Find the hyperedge nodes of a selected hyperedge (or set of hyperedges);
- Determine all hyperedges that share specific node (or set of nodes);
- Filter nodes/hyperedges utilizing topological features, such as node degree;
- Estimate the cardinality of hyperedges;
- Support editing of nodes and hyperedges if needed.

This thesis project is based on the previous research on the Onion approach. Many developments in information visualization have been realized since the first version of the Onion tool was implemented. Nevertheless, hypergraph analysis and visualization are still active fields of research with many possible domain applications. We believe that there is a possibility of further research of hypergraph visualization based on the Onion approach.



☐ Show grid lines
☑ Show cursor coordinates on status bar
☑ Show selected nodes on status bar
☑ Show selected edges on status bar
✓ Show status bar
Hide status bar
Auto-hide status bar

Node List	Edge List
n2:This could be a node label	e3:
n6:	e4:
n4:	e1:This could be a hyperedge label
n3:	e2:
n5:	e0:This could be a hyperedge label
n1:	e9:
	e8:
	e7:
	e6:
	e5:

selectedNodes: [n4, n3, n5, n1]   selectedEdges: [e1]   x = 445.0, y = 572.0

Figure 1.1: *Main window of the original version of the Onion tool. The hypergraph view is on the left-hand side, and the control panel is on the right. The small sample hypergraph has six nodes and ten hyperedges (four of them being ordinary edges with a cardinality of two). Edge* e1 *was selected by the user (highlighted in pink). Figure by Kerren and Jusufi [12].*

1.2 Motivation

Information visualization is the practice of turning unstructured bodies of complex data into visual form and actionable insights (as well as a research discipline focusing on such methods) [2, 3]. With regard to network data, people usually endow the investigated objects with pairwise relationships for both analysis and visualization [1]. However, in many real-world problems, relationships among our objects of interest are more complex than pairwise. Therefore, we can use the hypergraph model to handle the complicated relationships among the entities of interest.

After decades of information visualization research, hypergraphs have attracted increasing attention due to their flexibility and efficiency in modeling complex data relations. Hypergraphs have found success by generalizing regular graphs in many applications, such as clustering, classification, and prediction [25].

The Onion hypergraph visualization tool [12] was developed in 2013. It aims to address the problems of edge clutter, overlaps, and edge crossings. However, some issues are not addressed, for instance: 1) the edge bundling method for ordinary edges is not entirely supported; 2) the original research paper does not focus on hypergraph comparison; and 3) it does not provide concrete evidence about the interactive visualization approach's scalability in the Onion tool. So, we want to contribute to the hypergraph visualization research by building upon the original Onion tool study.

1.3 Problem Statement

According to the prior research on hypergraphs, we know that hypergraphs can provide a natural method to represent multi-way interactions. The Onion approach was shown to be promising for hypergraph visualization and interactive exploration, however, several challenges and opportunities for further improvements were identified. By addressing these issues and collecting evidence about the resulting approach via validation, we aim contribute to the ongoing research on hypergraph visualization.

The main research aim of the original Onion approach was to solve the issues of edge clutter, overlaps, and edge crossings of the hyperedges. Onion uses a radial layout approach to avoid edge clutter. The nodes of the hypergraph are evenly distributed on a virtual circle. However, the current representation of ordinary edges (i.e., edges connecting two vertices) in the center of the main view is arbitrary. We want to find an existing edge bundling method that is applicable for this view and facilitates hypergraph exploration tasks.

Another shortcoming of original approach is related to the lack of support for comparison of several hypergraphs. We should thus extend the previous implementation accordingly and collect evidence about the usability of hypergraph comparison functionality for the new version of Onion.

Furthermore, we do not know the precise boundary evidence of the Onion approach scalability concerning computational performance and usability, such as the maximum number of nodes and edges that Onion can represent without running into issues perceived by the users. Our analysis and evaluation will expose all the problems mentioned above.

In order to find answers to the respective research questions, we will implement an entirely new version of the Onion tool as a Web application. We expect that we will be able to answer the research questions after the study and that we will be able to provide evidence to prove the advantages and identifying the remaining shortcomings of the Onion approach.

1.4 Research Questions, Objectives, and Contributions

Our project investigates the properties of our new Onion hypergraph approach implementation compared with the old, original version of the Onion tool with regard to data representation and interaction aspects. To attain our research purpose, we define three research questions and related objectives presented below.

With this thesis project, we intend to address the research questions specified in Table 1.1, each of which is further discussed below.

Table 1.1: Thesis project research questions.

RQ1	Does the edge bundling approach implemented in the new version of Onion for binary edges facilitate hypergraph exploration?
RQ2	Can the interactive visualization approach implemented in the new version of Onion support the comparison of several hypergraphs effectively and efficiently?
RQ3	How does the interactive visualization approach implemented in the new version of Onion scale to larger hypergraphs with regard to computational performance and usability?

An edge (or a link) is an essential visual primitive mark for representing relational data in information visualization. Edges provide a visual encoding of relations between nodes in graph/network data and other scenarios (for example, highlighting relations between several similar or connected items in a scatter plot). However, visualizations frequently suffer from edge clutter issues when numbers of visible data items become large, as a mass of edges can overwhelm the performance, and hide underlying patterns [20]. Edge bundling has appeared as an essential technique for decreasing visual clutter in visualizations. With **RQ1**, we want to explore if edge bundling can improve the performance of the binary hyperedges (i.e., edges connecting only a pair of nodes) in Onion with regard to representation and exploration of hypergraph data.

With **RQ2**, we argue that comparison of several hypergraphs is an independent and important issue that should be addressed in our project. Data analysis often involves comparison of complex objects. According to the research of Gleicher et al. [26], better support for such comparisons helps with the increasing data amounts and complexity of data analysis, and it is thus sought after in interactive visual approaches. How to represent several hypergraphs for comparison purposes, how to highlight their similarities and differences for the user, and similar non-trivial design issues and their outcomes must be investigated in order to address this question. The aspects of effectiveness and efficiency mentioned in this question are related to the problems of usability and evaluation in visualization and human-computer interaction fields; *effectiveness* is associated with the accuracy and completeness of the solution supported by an interactive visualization approach (e.g., whether the users are able to solve their tasks with low error rates), while *efficiency* refers to the resources necessary to achieve the users' goals (e.g., whether the users are able to solve their tasks with low response times when using the interactive visualization tool) [27].

With **RQ3**, we investigate the scalability of visualization of hypergraph data. Visual scalability is the capability of visualization tools to display large datasets in terms of the number or the dimensionality of particular data elements [28]. Furthermore, higher-resolution displays are becoming accessible to more users as display technologies decrease in cost and software for the displays improves [29]. As we explore these opportunities to improve the scalability of the Onion tool, we use existing techniques to break both technical limitations (for instance, the issues of memory consumption and ability of the implementation to render the updated visualization and react on user interaction without critical delays) and usability limitations (for instance, the issues of visual clutter and perception of a larger number of nodes and edges).

We expect this thesis project to result in the contributions corresponding to the objectives formulated in Table 1.2.

Table 1.2: Thesis project objectives.

O1	Analyze the improvements of the new version of Onion with regard to visual representation and interaction capabilities
O2	Collect and analyze evidence of the usability of edge bundling in the context of hypergraph exploration tasks supported by the new version of Onion
O3	Collect and analyze evidence of the effectiveness and efficiency of a hypergraph comparison technique supported by the new version of Onion
O4	Collect and analyze evidence of the scalability of the interactive visualization approach implemented in the new version of Onion

As we mentioned before, our project is a further study of the Onion approach, and we try to address the challenges and opportunities for improvements identified in the original study of Kerren and Jusufi [12] mentioned in Section 1.1.4. The first steps of this thesis project will be concerned with the analysis of the relevant prior works, which will motivate the design and implementation efforts. By reflecting on the outcomes of the implementation and validation stages, we will reach the objective **O1**.

The research question **RQ1** is concerned with finding a solution for edge bundling that is not implemented in the previous approach implementation. The evidence necessary for answering this question will be collected as part of the work on **O1**, but also **O2**, more specifically, the questions and tasks involving edge bundling will be included in a user study that will form an important part of the validation of this work.

With our research question **RQ2**, we want to find an effective and efficient way to support the comparison of several hypergraphs. By achieving **O1** and **O3** (which means that the hypergraph comparison functionality will also be included in the evaluation of the new implementation), we will be able to answer this question.

Finally, the research question **RQ3** is related to the scalability of the visual approach. According to the original study [12], it was assumed that the original Onion approach would be able to scale up to hypergraphs with approximately 100 nodes and 150 hyperedges. By achieving **O1** and **O4** (with both the results of the user study and the results of case studies with real-world data), we intend to collect the sufficient evidence to address this assumption and answer the respective research question.

1.5 Scope and Limitations

This thesis project focuses on design, implementation, and evaluation of an interactive visual representation approach, and this work aims to solve the common problems of the hypergraph visual representation, such as the clutter, overlaps, or edge crossings, in the scope of the proposed implementation. The implementation will be evaluated through (1) case studies with real-world data and (2) a user study involving participants who have experience and relative knowledge of information visualization. However, to ensure that our study will not become a too broad topic and out of control, we limit our hypergraph study area and only focus on undirected/disordered hyperedges.

With regard to the implementation, we intend to use two different libraries, D3.js and React, to implement our hypergraph visualization as a web-based tool. However, since these two libraries do not have a perfect compatibility with each other, there are some implementation limitations that will not be addressed within the scope of this thesis project (these will be discussed in further detail in the respective section).

Finally, it should be mentioned that this thesis project does not focus on longitudinal studies of the Onion tool application beyond the scope of the case studies presented in this report.

1.6 Target Groups

As with most productive mathematical theories, the theory of hypergraphs has been widely used in various fields. Hypergraphs model many practical problems in many different sciences such as psychology, genetics, and various human activities [30]. Hypergraphs have an enormous potential capability as a tool to resolve optimization problems such as minimization of logical functions [31] or partitioning of a discrete system [32].

Our study might interest researchers who study hypergraph data analysis. Furthermore, researchers in information visualization and visual analytics may find the design concepts or results discussed in this report useful for their own work. One example is the ISOVIS group, which has initiated this thesis topic, motivated by the first version of the Onion tool. Other information visualization experts and students, who want to analyze their experimental data using the Onion tool or the evidence about the effectiveness and efficiency of Onion compared with their approaches, might also be interested in the results of this work.

1.7 Report Structure

The following Section 2 focuses on the methodology of this work: there, we define the problems we will solve and the solution methods we intend to use. In order to study the existing approaches and ideas applicable to our efforts, we conduct an analysis of the related work in Section 3. Afterwards, we dive into the issues of design and implementation of our visualization approach in Section 4. There, we discuss the main design ideas for our Onion tool. Then we will describe the technologies and design concepts we used behind the Onion tool in detail. This section will also proceed with a detailed discussion of the resulting implementation. Next, we conduct the evaluation of our approach in Section 5. First of all, we apply the tool to real-world data sets and discuss the outcomes of the respective case studies. Then we discuss a user study conducted with our approach; afterwards, we look at the collected opinions from our survey, find out the better potential methods related to our visualization, and discuss additional aspects of the behavior of our implementation. After the analysis is finished, in the final Section 6, we summarize our evidence to answer the research questions we defined for this thesis project. We discuss if our study could meet the research questions and mention other learning directions that we can select to continue working on this topic.

2 Method

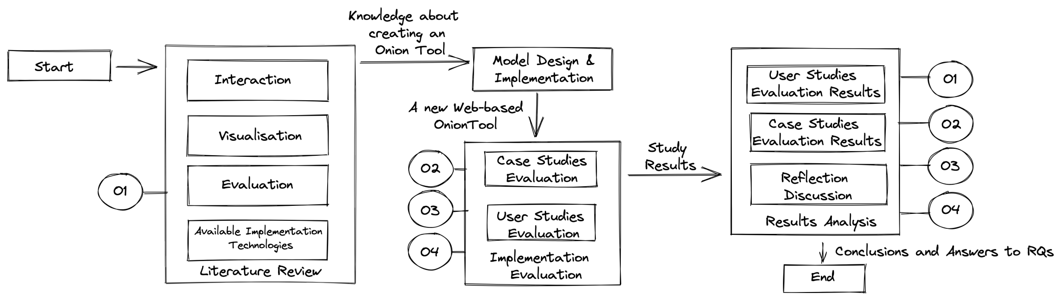
To build up an integrated study process to solve our problems, defined in Section 1.4, we follow a structured methodology explained in this section. The study method includes four steps: Literature Review, Model Implementation, Evaluation of the Implementation, and Results Analysis.

2.1 Scientific Approach

The approach taken in this work follows the overall process of designing and validating novel information visualization techniques, as discussed by Munzner [33], with a focus on collecting empirical evidence about the properties of the implemented approach via evaluation, as discussed by Purchase [34] or Elmqvist and Yi [35], for instance. The research methods used in this project thus include the analysis of related work, the analysis of requirements and design alternatives, case studies, user-based evaluation, and critical discussion. With regard to the analysis of the collected empirical data, we mainly focus on qualitative methods.

2.2 Method Description

In this subsection, we discuss the main steps of our approach in detail. Based on the research questions and objectives mentioned in Section 1.4, we design the methodology for our study. Figure 2.1 shows the interconnection of the methodological steps and the objectives from each step.



Start
Interaction
Visualisation
Evaluation
Available Implementation Technologies
Literature Review
O1
Knowledge about creating an Onion Tool
Model Design & Implementation
A new Web-based OnionTool
O2
O3
O4
Case Studies Evaluation
User Studies Evaluation
Implementation Evaluation
Study Results
User Studies Evaluation Results
Case Studies Evaluation Results
Reflection Discussion
Results Analysis
O1
O2
O3
O4
Conclusions and Answers to RQs
End

Figure 2.1: *An overview of the methodological steps and outcomes planned for this thesis project.*

2.2.1 Literature Review

To answer the research questions for this project, we need to have enough knowledge about building an effective and efficient hypergraph visualization tool, thus we start with an analysis of related work. The literature study includes deciding on the sources of information, retrieving and studying the respective scientific and technical articles, and discussing the corresponding methods and techniques. Please note that we do not pursue a rigorous systematic literature review [36] process here, but rather focus on the literature that is relevant for our research questions and objectives. We look for the entry point based on the following four points: visualization interaction, visual evaluation, visual analysis of hypergraphs, and available visualization implementation technologies. Thus we start our inquiry by using the databases of the LNU library to search for different types of references from reputable sources, using the respective keywords such as “hypergraph visualization”. Furthermore, we inquire information about methods and algorithms for edge bundling, hypergraph comparison, and visual interaction, using the respective keywords. Besides searching for such publications directly, we also consider following relevant references from the found publications (similar to the “snowballing” approach [36]), including the original study by Kerren and Jusufi [12]. Collection of knowledge from the literature review will be part of direct work on **O1**, as it will affect the design choices for the implementation (the other objectives and research questions will be affected indirectly).

2.2.2 Model Design and Implementation

Here, we apply the existing methods to our hypergraph visualization model based on the knowledge obtained from the literature review. Following the Onion tool design of Kerren and Jusufi [12], we will build an entirely new web-based Onion tool using JavaScript. New algorithms and visual design ideas may be required during the development, while keeping the research questions and objectives in mind. Moreover, finding the solution will be discussed with the supervisors and probably with other specialists from the field. This part of the project work will indirectly contribute to the objectives.

2.2.3 Evaluation of the Implementation

Evaluation of the implemented tool is a necessary process for the information visualization project during our hypergraph visualization model development. Moreover, collecting evidence for **O2**, **O3**, and **O4** in the evaluation section is essential for validating our hypotheses. As motivated by the existing work on evaluating visualization approaches [35,37,38] and discussed in the respective part of the literature review, we plan to use several validation approaches. First, we apply our implementation to real-world data to test its feasibility and support for arriving at relevant findings and insights, which will take the form of case studies. Then we conduct a user study with multiple participants to test the usability of our approach for several analytical tasks, with the subsequent analysis of effectiveness and efficiency [27]. We also collect user feedback necessary to answer several other research questions as part of this step.

2.2.4 Results Analysis

As the outcome of the previous step, we will collect the empirical data as part of the user study. While some of the results might be available in quantitative form, we do not expect the number of participants to be sufficient for obtaining data sets suitable for in-depth statistical analyses, thus, we will mainly focus on descriptive analyses of the collected quantitative data, and otherwise, rely mainly on qualitative methods. The feedback received from the users will also be analyzed using qualitative methods. The analysis of evaluation results will help us finally achieve the objectives **O1–O4** and thus answer the research questions **RQ1–RQ3**. Furthermore, the outcomes of results and discussion will allow us to share key findings and recommendations with the users, also and share our hypergraph visualization model’s mechanism and good practice with other experts from the field.

2.3 Reliability and Validity

Reliability and validity are the vital features that can make the reader consider our project results convincing and definitive.

Reliability of the approach is related to the concerns of reproducibility (i.e., obtaining consistent computational results) and replicability (i.e., obtaining consistent results of a complete study), as discussed by Fekete and Freire [39]. Regarding the reproducibility in visualization information, the others who want to reproduce our particular visualization results can quickly get a similar visual illustration using our approach with the same data set and same parameters configurable via the user interface, as most of the implementation aspects will rely either on deterministic algorithms, or on stochastic algorithms with pre-defined random number generator seeds, e.g., for edge bundling purposes [20]. However, regarding the case study findings and user study results within the implementation evaluation stage, several factors might cause the same method to create different output assessments. To replicate the evaluation result as in this work, the examiners who attend the implementation evaluation should have a related background and experience in information visualization. Furthermore, the examiners need to go through the same implementation evaluation process with the same data set as this work. The details of the implementation evaluation can be found in the the respective section.

When viewing the validity of this project, we ensure to analyze as many aspects as possible regarding the evaluation of this work. To reduce problems with construct validity, we assure having a concrete reference and comparison as our evidence to support our theoretical assumptions.

In the context of information visualization, internal validity is associated with the confidence that the outcomes of the study are affected by the controlled parameters and are not distorted by other confounding factors, as discussed by Elmqvist and Yi [35]. Considering the internal validity, we explain the steps that produced a particular result in the respective part of the report. Moreover, the previous study of the Onion tool can also help us design the respective evaluation steps and avoid the internal validity problems.

External validity is a relatively common problem in studies, which is related to the degree with which the lab findings can be generalized to further data and problems, for instance, for real-world applications [35].

To address this concern, we apply our approach to several real-world data sets from various domains as part of the case studies; and furthermore, besides testing the implementation ourselves as the users, we collect and analyze the feedback on usability as part of the user study with several knowledgeable participants.

2.4 Ethical Considerations

First of all, we should note that the implemented visualization approach is data-agnostic as long as the provided input files conform to the established GraphML [23] format for hypergraph data, as discussed in the following sections; thus, this approach is not associated with any particular applications initiated by the users with their own data sets.

While the current prototype implementation is implemented as a web-based tool, it does not make use of any backend components or persistent data storage, and thus no personal information of the respective users is collected or stored. The materials used for the case studies were also based on publicly available data from external sources, as discussed in the respective sections.

With regard to the user study conducted as part of the validation of this project, a procedure similar to the typical human-computer interaction study protocol [34] was followed. The participants were informed that their involvement was voluntary and could be stopped at any moment, and that only their feedback, responses to the study tasks, and post-study questionnaire results would be used, with no personal information recorded or used afterwards.

3 Related Work

In this section, we discuss the important prior work relevant to the problem addressed in this thesis project. The related work includes literature on hypergraph data structures (used for the data modelling for our work), existing representation and interaction approaches in information visualization in general (establishing the basis of our proposed approach), edge bundling techniques (as required for achieving one of the objectives of this project), and existing interactive visual analysis approaches focusing on hypergraphs in particular (in order to allow us compare our proposed approach to the existing solutions).

3.1 Hypergraphs

In the past few years, the emergence of larger and more complex data sets in various academic, industrial, and public settings has driven a new larger demand for representation and analysis of more complex graph/network data structures than had been employed previously.

In general, graphs are combinatorial models for representing relationships between particular objects. Hypergraphs provide a natural way that encodes relationships between a arbitrary number of objects/entities in data, i.e., more than two objects at a time could be part of the same relationship. Hypergraphs thus can be related to and applied in scenarios involving sets as data structures, e.g., for representing membership of several objects in several sets simultaneously, while being able to use the terminology and most of the methods from graph/network analysis.

Hypergraphs have been recognized as a viable and useful data model in several application fields such as computer vision [40], bioinformatics [41], and information retrieval [42].

As we mentioned in Section 1.1.1, hypergraphs generalize simple, ordinary graphs by allowing edges to be incident to multiple objects, resulting in so-called hyperedges. In this thesis project, we only discuss hypergraphs with undirected hyperedges. Undirected hypergraphs can help model such problems and data sources as satisfiability problems [43] or databases [44]. Besides, undirected hypergraphs have been widely used in machine learning tasks as the data model and classifier regularization [45]. As a simple definition, simple undirected graphs can be viewed as an exceptional case of hypergraphs, in which every hyperedge contains two nodes only [41]. In mathematics, an undirected hypergraph H can be defined as a pair (V, E), where V is a set of vertices, and E is a set of hyperedges between the vertices; thus, each hyperedge is a set of vertices: $E \subseteq ((u, v, \dots) \in 2V)$ [46].

As shown in Figure 3.1, the work by Klamt et al. [41] illustrates an excellent example to explain the fundamental concepts of an undirected hypergraph by using the protein-protein interaction networks.

There is a large amount of literature on hypergraph partitioning, which arises from various practical problems, such as partitioning circuit netlists [47], categorial clustering data [42], and image segmentation [48]. Besides, hypergraph learning can be seen as passing information along with the hypergraph structure in analyzing the structured data and solving node classification [49], link prediction [50], and community detection [51] problems. Hypergraph learning models investigate the high-order correspondence among data, which leads to a noble capability of association modeling in practice.

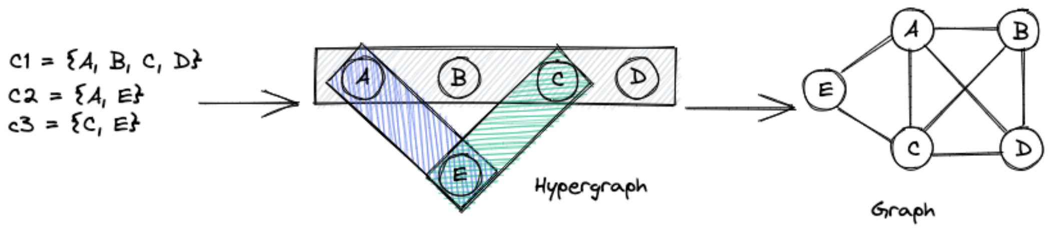


Figure 3.1: *An example to explain the fundamental concepts of an undirected hypergraph by using the protein-protein interaction networks. Based on the work by Klamt et al. [41].*

3.2 Representation and Interaction in Information Visualization

As discussed in Section 1, information visualization focuses on interactive visual representation of abstract data that can facilitate the users' exploratory and analytical tasks [2]. The information visualization reference model is an example of a reference model for dealing with visualization information, introduced in similar forms by Chi [52] and Card et al. [2]. The reference model successfully modeled a wide array of visualization applications and was functionally equivalent to the data flow model used in existing graphics toolkits such as VTK [53]. The information visualization reference model defines three main stages: data tables, visual structures, and views.

- Data tables: relations (cases by variables) + metadata
- Visual structures: spatial substrates (i.e., layout regions) + visual elements + graphical properties
- Views: graphical parameters (scaling, zooming, clipping, etc.)

Furthermore, these particular components end up in an interactive representation presented to the user, who can adjust all of them iteratively via various interactions. As shown in Figure 3.2, initially, raw data with a specific idiosyncratic format is loaded and then represented as structured data tables to generate the desired information. Then data tables are manipulated and transformed into one or more visual representations. The end-user conducts and interacts with the visual representation in one or more views at the last step. One of the main benefits of the information visualization reference model is that it explicitly represents interactions. Moreover, it provides a general template for structuring visualization applications that separate data and visual models to enable multiple visualizations of a data source, particular visual models from displays to boost various visualization views, and use modular controllers to handle user input flexibly and reusable fashion.

The particular purposes and possible interactions to be supported by a visualization approach are typically established according to the specific user tasks, i.e., functional (and in some cases non-functional) requirements for the respective approaches. Multiple taxonomies or typologies of such user tasks exist in information visualization and visual analytics literature. One of the classic taxonomies was introduced by Shneiderman [24], who formulated the so-called visual information seeking mantra: "Overview first, zoom and filter, then details-on-demand". Shneiderman's mantra is an influential organizing principle for the creation of visualization systems [54]. It addresses several steps. Firstly, in the overview step, the

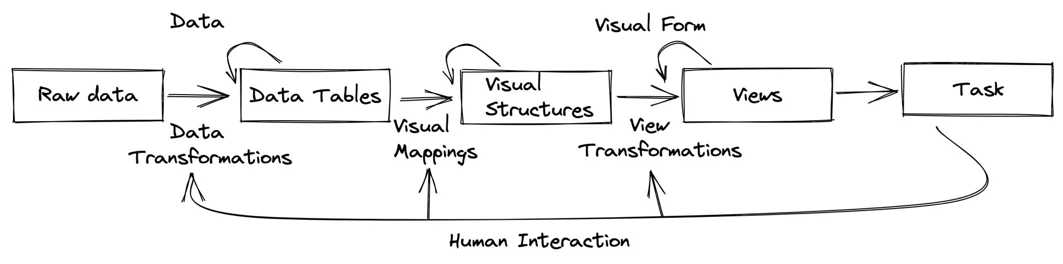


Figure 3.2: *The information visualization reference model. Based on the work by Card et al. [2].*

software system needs to interpret the entire data set and present it visually in a suitable way, which brings a quick understanding to the users and provides context to the next steps in the visualization. Secondly, zoom and filter are the interaction techniques that allow the user focus on subsets of interest in the data. The user can zoom into each attractive container in turn and filter extraneous data based on the desired attributes. Finally, the details-on-demand technique gives the user control of the data with further exploration. The most common implementation of details-on-demand is a tooltip. The tooltip presents more details about a particular data point as the user hovers the mouse pointer over the respective visual item. Furthermore, it allows the user to peruse data and gather insight at their leisure dynamically, without cluttering the main view.

As we mentioned in Section 1.1.3, scalability is an open problem with multiple facets in the study field of information visualization, including massive networks, huge multi-dimensional heterogeneous data sets, and complex data streams. Besides, this issue also leads to the question of the perceptual scalability of visualizations. Hence, we apply the information visualization reference model and Shneiderman's mantra model for the Onion's visualization interaction implementation regarding the above problems, as the design of the visual representation on its own can be insufficient without proper support from interactions.

Based on the information visualization reference model, we first implement a basic version of the Onion tool based on the data format and small-scale sample raw data offered by the ISOVIS group. We also play a role as a user in iterative testing of the information visualization reference model to find the answer of scalability in larger hypergraphs regarding computational performance and usability in the new implementation of Onion. Exporting the displayed graph in different image formats and as a GraphML file is also possible if the user edited the graph itself. Besides, Shneiderman's mantra offers a standard principle of visualization interaction. Moreover, it brings up the essential features that facilitate the analysis process.

3.3 Evaluation Approaches and Performance Measures for Information Visualization

Evaluation of a visualization technique is an important activity that can provide evidence about the usability of a new technique, including such aspects as effectiveness and efficiency [27], for instance. Performance and preference measures are widely used in the assessment of visualization techniques [35, 37, 38].

The graphs, for instance, are often visualized as the node-link diagram, where nodes are depicted as points and edges as line segments connecting the corresponding points. Multiple studies examine the readability of node-link diagrams. For instance, the studies by Purchase [55] and Purchase et al. [56] examine how the drawing aesthetics such as edge crossing and display of symmetries influence performance for graph visualizations. Several studies conducted by Huang et al. [57–60] focus on testing the optical network perception by using eye-tracking.

From the above studies, we understand that the performance depends on the particular graph layout. In terms of path searching tasks, the edges alongside the paths of nodes affect drawing's readability and trigger extra eye movements, and the edge crossings create confusion during reading. Furthermore, these studies performed several practical tests on the human understanding of various aesthetics criteria. The aesthetics considered were: bends, edge crossings, minimum angles, orthogonality, and symmetry. The experimental results confirmed that conditions relying on symmetry took significantly less time than the minimum angle and orthogonality with regard to task completion time, suggesting that symmetry has a significant positive effect than the other aesthetics when it is at a maximum value.

The proposed Onion tool uses a radial layout approach. A radial layout [6, 33] makes the center of the circle model compactly displaying summary statistics or indicating points of interest. We believe that a radial layout can effectively reduce eye movements compared with a matrix arrangement. Besides, the symmetry studies inspire us in using the mirror image layout when two hypergraphs have been imported. This becomes important as we consider support for comparison [26] of several hypergraphs, which constitutes one of the objectives of this thesis project. To collect the respective evidence as part of the evaluation, we will include tasks related to comparison of several hypergraphs for our validation approaches, so that we can investigate if the user can quickly find out the difference between two hypergraphs with the mirror image layout.

Different types of evaluations have been proposed and applied to assess the visualization approaches, ranging from controlled experiments to longitudinal studies [35, 37, 38]. Nevertheless, the choice of an evaluation strategy and design of a particular evaluation study is a difficult and non-trivial issue. One example is evaluating a visualization's utility to measure accuracy and time in a study where participants perform benchmark tasks. Such an approach can provide particular quantitative measurements, but it would not be adequate for assessing visualization's more nuanced expressive power [61].

To understand the feasibility of a proposed interactive visual analysis approach for real-world data, a case study can be conducted by the authors of the respective technique, sometimes in collaboration with domain experts [38]. However, one could argue that a case study focuses more on the data-oriented outcomes and insights rather than the perceived usability of the visualization itself. Thus, while we intend to conduct case studies, we complement them with another user-centered evaluation approach. Based on the work by Stasko [62] that argues for multi-faceted evaluation of the *value* of visualization, *ICE-T*, a heuristic evaluation methodology, has been proposed by Wall et al. [61]. The title of ICE-T is an abbreviation of four concepts, *Insight*, *Confidence*, *Essence*, and *Time*. The purpose of these evaluation criteria is the following [61]:

- *Insight*: A visualization's ability to spur and discover insights or insightful questions about the data.

- *Confidence*: A visualization's ability to generate confidence, knowledge, and trust about the data, its domain, and context.
- *Essence*: A visualization's ability to convey an overall essence or take away the sense of the data.
- *Time*: A visualization's ability to minimize the total time needed to answer a wide variety of questions about the data.

ICE-T thus defines the particular criteria that multiple raters can understand and apply with consistency. The approach itself uses a questionnaire form with questions/statements corresponding to the four concepts discussed above (for example, "The visualization exposes individual data cases and their attributes"), which the raters reply to using a 7-point Likert scale [63] (from "Strongly disagree" to "Strongly agree") or skip the question. The results can afterwards be quantified, averaged over multiple raters, and aggregated over the respective criteria. ICE-T thus provides an effective method to evaluate the visualization, it is practical to implement with even a small number of study participants (which is another important consideration for the scope of this thesis project), and it helps pinpoint some of the shortages of the visualization. It is meaningful and able to evaluate research prototypes of visualization applications. ICE-T has been successfully applied in a range of visualization research studies recently, including the evaluation of multivariate network visualization techniques by Nobre et al. [64]. Thus, we have chosen this approach for conducting a user study of our interactive hypergraph visualization approach as part of this project.

3.4 Edge Bundling Approaches

Node-link visualization approaches generally suffer from visual clutter induced by many edge crossings and node-edge overlaps when the visualization contains an enormous number of nodes and edges. This problem can easily overwhelm the users and obscure the underlying data patterns. Moreover, it becomes even more critical when nodes' positions are fixed. Edge bundling techniques can help to alleviate these issues by visually clustering edges along comparable routes. Many edge-bundling methods are introduced in the literature, e.g., the survey by Zhou et al. [20] discusses several groups of approaches, such as hierarchical edge bundling and geometry-based edge bundling. Hierarchical edge bundling methods [65] are used to visualize graphs that contain a hierarchical structure. Geometry-based edge bundling algorithms [66] search for the plane for the configuration of each edge in discretization grids of the visualization plane.

Compared with these approaches, force-directed edge bundling [67] does not require the graph to contain a hierarchy architecture and no control mesh. On the contrary, it follows a self-organizing approach to bundling in which edges are modeled as flexible springs that can attract each other.

Most of the graphs are generally visualized as node-link diagrams, in which dots depict the nodes, joined by lines or curves for the edges [67]. A straightforward way to allow groups of edges to be merged and drawn together in the graph would provide a clean and disordered layout graph and efficiently handle the uncluttered problem when graphs comprise many nodes and edges [65]. One standard bundling method is hierarchical edge bundling, as mentioned above. However, it is not trivial to create a suitable hierarchical edge bundling for a general graph. This issue motivates us to use a self-organizing, force-directed approach. The behavior of force-directed edge bundling is easy to understand because of the straightforward physical model; the core algorithm can be implemented in a few lines of code and can quickly be extended to accommodate additional layout models. As the Figure 3.3 shows, in each iteration step, each subdivision point updates its position by moving a small distance toward the direction of the combined force Fp_i. For a subdivision point p_i on edge P, the combined force Fp_i exerted on this point is a combination of the two neighboring spring forces Fs exerted by p_{i-1} and p_{i+1}, and the sum of all electrostatic forces Fe [67].

To solve the edge crossing problems mentioned in Section 1.2, we have thus implemented a force-directed edge bundling approach for binary/ordinary edges in our Onion implementation.

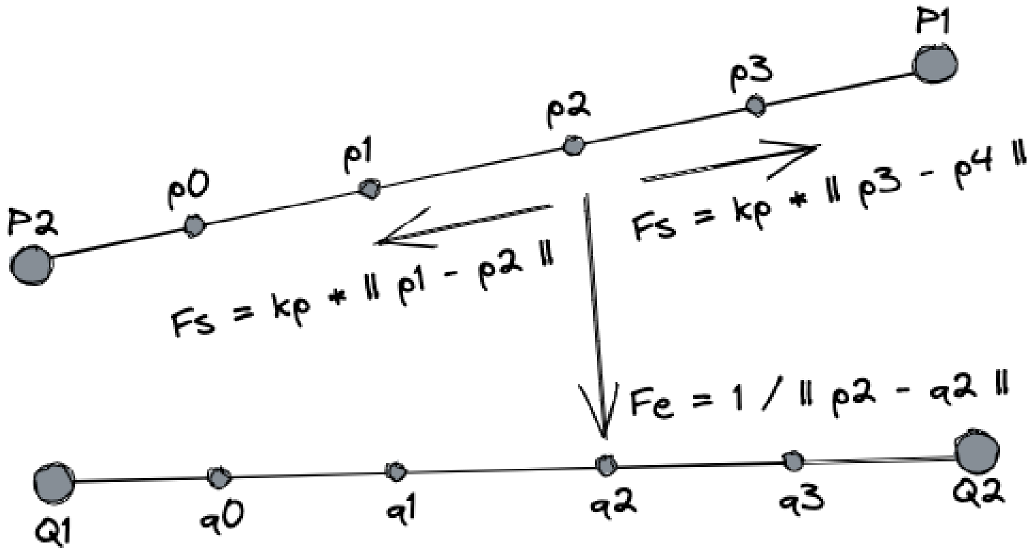


Figure 3.3: *Two interacting edges P and Q considered for the force-directed edge bundling approach. The spring forces Fs and the electrostatic force Fe that are exerted on subdivision point p_2 by p_1, p_3, and q_2 are shown. Based on the work by Holten et al. [67].*

3.5 Interactive Analysis and Visualization Approaches for Hypergraphs

As mentioned in Section 1.1.4, the Onion tool is a novel radial visualization approach for undirected hypergraphs that has been developed by Prof. Dr. Kerren and Dr. Jusufi in 2013 [12]. Many current network visualization tools handle hypergraphs by extending standard graph layouts with vertices and edges. Cluttering, overlapping, or numerous edge crossings are the typical issues encountered in the hypergraph visualization research. However, standard graph layouts are hard to render or scale with larger hypergraphs. Besides, there are still many problems with standard hypergraph layouts stopping users from investigating the data with a deeper perspective, such as data editing, filtering, comparison, and data statistics. The Onion tool aimed to solve these issues for undirected hypergraphs. Our approach discussed in this work extends the original Onion approach and aims to address the shortcomings and opportunities for improvements identified previously.

Besides the original Onion approach, we should also analyze other existing approaches for hypergraph visualization and analysis. One basic approach would be to introduce a number of extra edges to replace a single hyperedge, which leads to additional clutter and loss of context for the users. Drawing a hypergraph as a bipartite graph [1] adds nodes to describe hyperedges and extra edges to attach them to the actual vertices [68], which also potentially increases visual complexity and causes confusion for the users. Other strategies for representation of hypergraphs can also be considered when thinking of each hyperedge as a set of nodes. Alsallakh et al. [69] discuss the existing set visualization approaches in their survey. Some of these techniques have been applied to hypergraph data, including Kelp Diagrams by Dinkla et al. [70] or variations of node-link diagrams in EGAN by Paquette and Tokuyasu [71]. However, those visual approaches exceed the available options for visual encoding (e.g., unique color hues necessary for visual marks) after one or two sets of hyperedges. Edge crossings and issues of clutter are also an issue here, e.g., the force-based hypergraph drawing approach by Arafat and Bressan [72] represents hyperedges as enclosing curves around sets of respective nodes—when a node belongs to several hyperedges, cluttering issues arise.

A different strategy would be to rely on a custom layout in order to represent hypergraph/hyperedge data. Hyper-Matrix by Fischer et al. [73] and the Parallel Aggregated Ordered Hypergraph (PAOH) approach by Valdivia et al. [68] use a linear matrix layout to avoid cluttering problem. They display elements like columns and hyperedges as rows. Furthermore, both of these approaches make use of the timeline concept, as they are dealing with dynamic or temporal hypergraph data. Compared with the other hypergraph graphs, we should note that while such a matrix representation has benefits regarding edge crossing issues, it can become problematic when considering restricted display space for larger hypergraphs. However, the literature review showed that these novel temporal hypergraph visualization approaches aim to support a broad set of interaction features helping users explore the hypergraph data. We shall use this observation as a guideline when designing and implementing a new version of the Onion approach, as discussed in the following section.

4 Design and Implementation

This section discusses the Onion tool in this project, including the steps, tools, techniques, and technologies we used. These design and implementation steps will be necessary in order to proceed with finding out the evidence answering our research questions. We divide this section into several parts. First, we provide a high-level overview of our Onion tool design. Next, we address the issues related to the supported data formats and models, and the details of the implementation stack used. Afterwards, we proceed with a detailed discussion of the functionality of our implementation.

4.1 High-Level Design Concerns

This section will describe a general idea about the Onion approach implemented in this project.

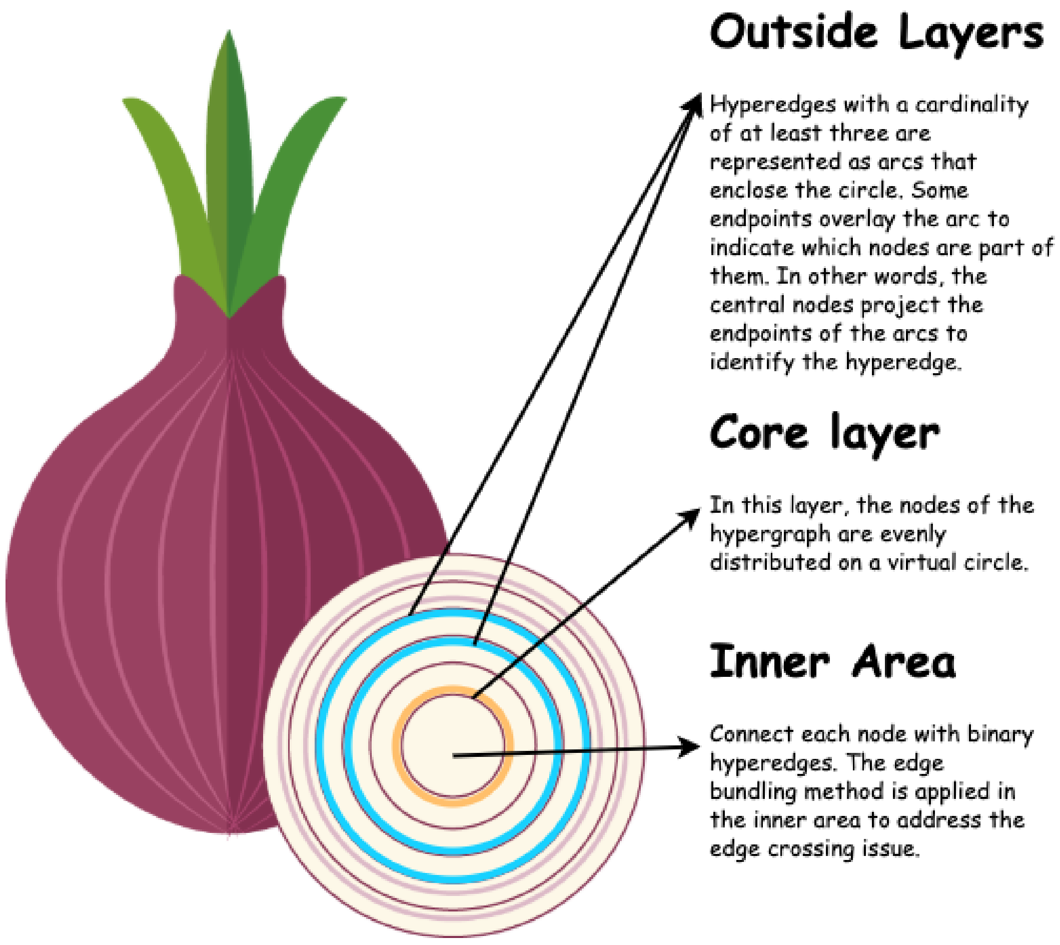

Outside Layers
Hyperedges with a cardinality of at least three are represented as arcs that enclose the circle. Some endpoints overlay the arc to indicate which nodes are part of them. In other words, the central nodes project the endpoints of the arcs to identify the hyperedge.
Core layer
In this layer, the nodes of the hypergraph are evenly distributed on a virtual circle.
Inner Area
Connect each node with binary hyperedges. The edge bundling method is applied in the inner area to address the edge crossing issue.

Figure 4.1: *The main visual metaphor used for the Onion approach.*

4.1.1 Main Visual Metaphor

As the Figure 4.1 shows, we layout the hypergraph on a circle to solve the clutter and many edge crossings, which happens in primarily temporal hypergraph tools. The metaphor used is of a common bulb onion vegetable, which is the source of the title of our approach. A bulb onion is formed by several outspread layers, as we can see if we cut it. Our visualization approach follows the similar principle, with multiple concentric layers laid out around the inner area, and all the layers being dependent heavily on the core layer. The nodes of a hypergraph are represented with larger dot marks located on the core layer. To represent their incidence to hyperedges, additional endpoint marks are displayed for each node, following a radial layout principle. Thus, there is an endpoint for each node–hyperedge combination present in the hypergraph data. Ordinary edges (i.e., hyperedges with cardinality of 2) are represented by links drawn in the inner (central) area instead. This design is the same as the original Onion approach described by Kerren and Jusufi [12].

4.1.2 Intended Workflow and Functionality

To understand the process of using the Onion tool from the user's perspective, we draw a flow chart below that describes the respective workflow (see Figure 4.2). To proceed with the improvements and collection of evidence regarding the Onion approach, we first need to build a similar implementation as the previous study, with the requirements and improvements mentioned in Section 1.1.4 and Section 1.3, respectively. Our plan is to implement a Web Onion version with a standard interaction, such as zooming, filtering, or recording the hypergraph elements, by using React and D3.js. We assume that these libraries can offer a suitable web-based platform while maintaining considerable performance, allowing us to eventually reach the objectives **O1** and **O4** (see Section 1.4). To specify the input hypergraph file(s), the new Onion tool only accepts the GraphML format [23] for the time being.

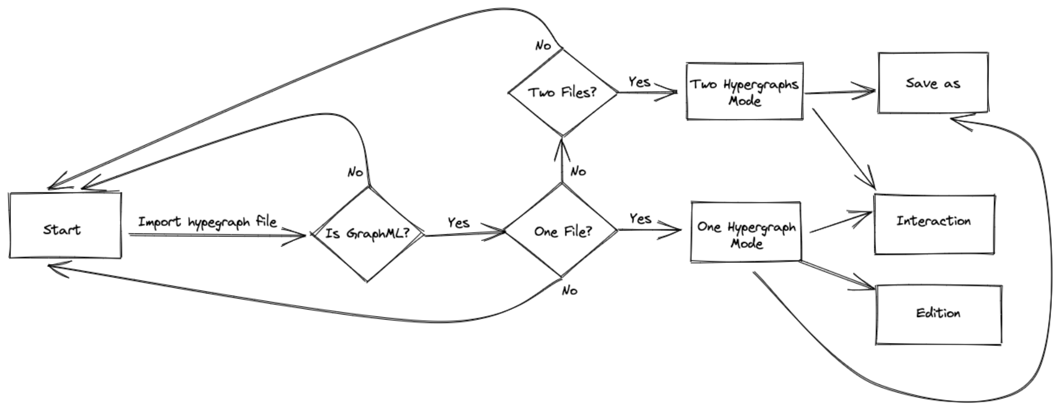


Figure 4.2: *Interactive hypergraph exploration workflow with the Onion tool.*

To address the previously identified challenges and suggestions [12], we have added several improvements to this project implementation. First of all, we implement the related mode helping users to determine all hyperedges that share specific nodes. Moreover, our plan is to implement two visual modes assisting the user in gaining information from the hypergraph model: cardinality mode for hyperedges' cardinality propose and sorting mode for efficient data interpretation. Furthermore, our plan is to implement filtering and editing methods to allow users analyze and change the topology of the hypergraph data according to their needs. However, due to the communication limitation between React and D3.js libraries, the nodes/hyperedges editing method is only supported with a single hypergraph model currently. These improvements should help us fulfill the objective **O1** and will also facilitate the tool use when collecting further evidence.

Regarding further improvements, we should note that finding a solution to the hyperedge crossing is one of the Onion tool's core design principles. Since the new Onion approach implementation follows the same architecture as the original one, hyperedges do not overlap with others in the outside layers. However, the issue of edge crossing within the inner/central area—where the ordinary edges with cardinality of 2 are drawn as arcs/links—should be addressed. Thus, in order to fulfill the objective **O2** and answer the research question **RQ1**, we use force-directed edge bundling for the central area of the Onion representation.

Another important task that should be supported by the new Onion workflow is comparison of two hypergraph files, as demonstrated in Figure 4.2. In order to reach the objective **O3** and answer **RQ2**, we implement a visual comparison mode for our approach inspired by a mirror image metaphor.

Moreover, as we mentioned in Section 1.1.3, scalability is a generally open problem within information visualization, and the scalability of Onion is also in question, as discussed previously. To show further details of a visualization to the user in limited screen space, we want the details of the hypergraph to be hidden when the user does not need them. To achieve the above demand, we position the control panel in a navigation drawer, which will only be displayed on demand on the left side of the screen. In addition, with the single hypergraph model, the editing functionality of the Onion tool is displayed by pressing a floating button. These improvements should help us reach **O4** and finally answer **RQ3**.

4.2 Data Model

Similar to other hypergraph tools, the Onion approach deals with graph/network data that describes relations between entities. Motivated by the original implementation [12], we use GraphML [23], an XML format for graph structures. GraphML is a format that widely accepts and suits particular data in graph drawing, editing, and storage. Additionally, it is simple to parse and understand for both humans and computers. Users can also extend the format in a well-defined way to represent additional data, while the data stored in GraphML can be easily identified and extracted by various visualization systems [23]. One of the principles of GraphML is to separate various layers of information conceptually (see Figure 4.3), such as graph structure, application data, topology, geometry, or graphics [23].

A hyperedge in the hypergraph model can connect any number of vertices [74]. A valid GraphML file that can be used to represent such data comprises at least one of the <*node*> elements (see an example in Figure 4.4). Furthermore, the <*hyperedge*> element contains any number of <*endpoint*> elements, which, in turn, refer to unique <*node*> elements. Hyperedges are the generalization of edges, and ordinary edges could hence be represented as hyperedges [23]. As Figure 4.4 shows, when considering two different <*hypergraph*> elements, the parser can immediately distinguish between them based on the incident endpoints/nodes. Besides, to allow more information for the parser to analyze, all elements in the data can use the XML attribute *description* to interpret the elements' details of the hypergraph, such as node or edge labels. The data used with the original Onion implementation is also provided by the ISOVIS group in GraphML format.

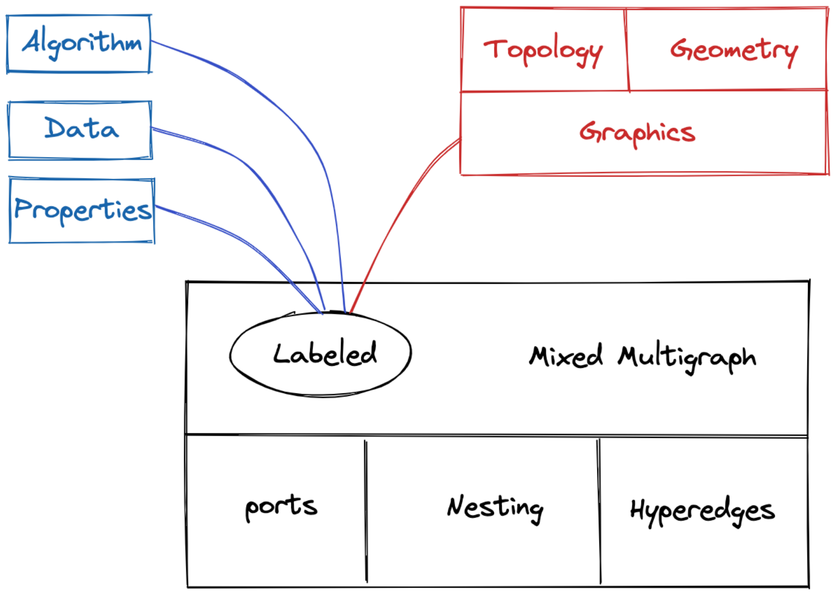
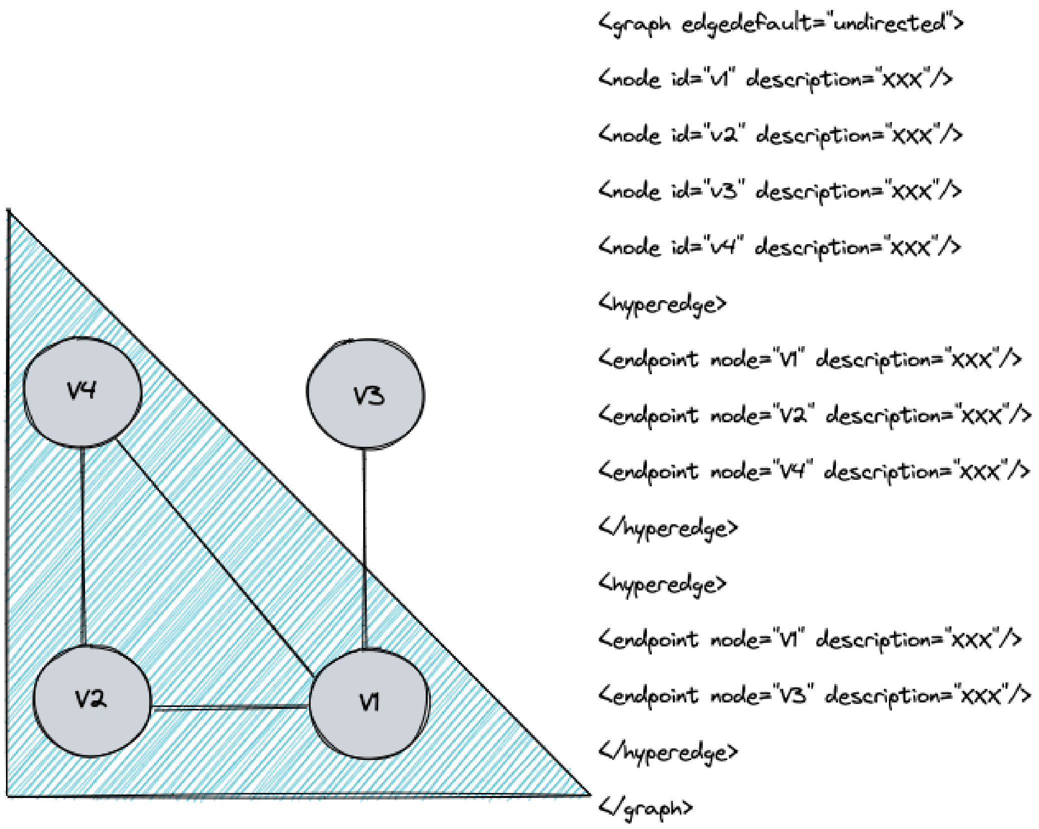


Figure 4.3: *The design of the GraphML format. The basic graph model of GraphML is labeled mixed multigraphs with optional node ports, hyperedges, and nesting. Graph drawing information is planned to be divided into topological and geometric, with a graphics layer on top. Like any other associated data, it will be encapsulated in a unique tag. Based on the work by Brandes et al. [23].*



```
<graph edgedefault="undirected">
<node id="V1" description="XXX"/>
<node id="V2" description="XXX"/>
<node id="V3" description="XXX"/>
<node id="V4" description="XXX"/>
<hyperedge>
<endpoint node="V1" description="XXX"/>
<endpoint node="V2" description="XXX"/>
<endpoint node="V4" description="XXX"/>
</hyperedge>
<hyperedge>
<endpoint node="V1" description="XXX"/>
<endpoint node="V3" description="XXX"/>
</hyperedge>
</graph>
```

Figure 4.4: *Example hypergraph in the GraphML format. The shaded area in the drawing represents a hyperedge with three vertices:* v1, v2, *and* v4.

4.3 Implementation Stack

React The Internet (World Wide Web) has become an integral part of society, and the web has evolved from a theoretical concept to a daily part of human life [75]. When a growing technology such as the Web is first introduced, Web development is driven by businesses demanding to build Web-based applications stable and efficiently. React is born for the above request. React was created with a single focus: to develop components for Web application frontends. According to the 2019 StackOverflow survey [1], React had a high level of popularity among the developers compared with most other Web development libraries. In other words, React is a mature and well-known technology that has good support from the framework developers and the user communities. It can save time for us to develop some essential functions but only focus on the Onion tool designing. React's API is straightforward to learn. Furthermore, React is being developed by Facebook and has further widespread community support. Considering that React is the one we have the most practical experience for Web development and contains all the tools we need to implement a Web-based tool for this experiment, we prefer it over other Web development libraries.

D3.js In the 21st century, human society is drowning in data, but starved of adequate tools for extracting essential information [76]. Because of the excellent human visual neural system, people rely on their sight more than anything else [77]. That is also why presenting data in a diagram can be more effective than in plain numbers.To make raw data easily analyzed, we need a framework to transform the data into visualization, and D3.js comes into play. D3.js is a JavaScript library for visualizing data using web standards. D3.js efficiently manipulates documents based on data. And it avoids exclusive representation and affords remarkable versatility exposing the full capabilities such as HTML, SVG, and CSS.

Based on our experience and the above considerations, we use React for our Web development and D3.js for this project's visualization implementation.

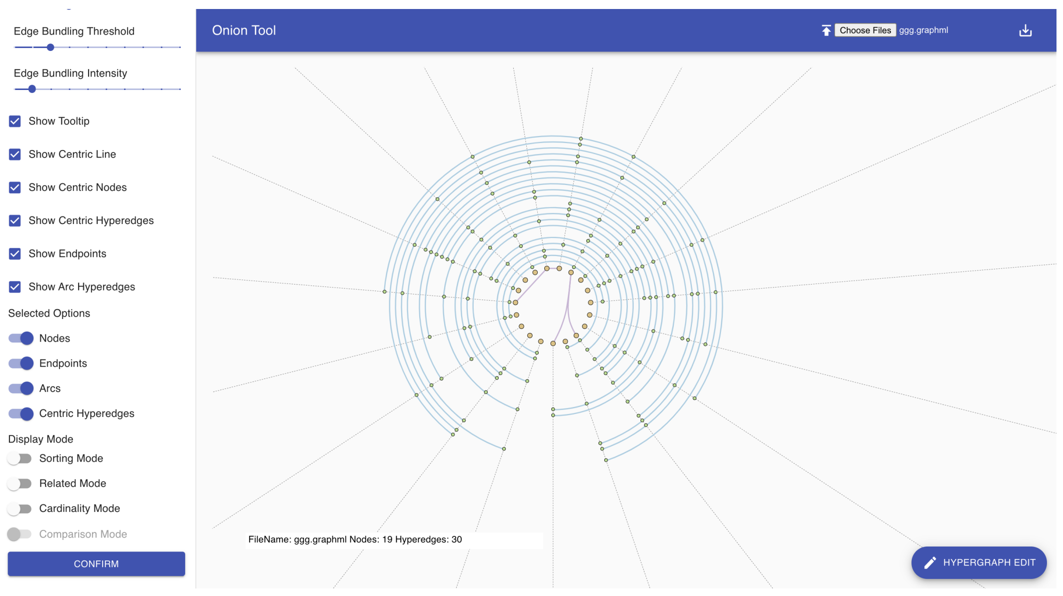
4.4 Visualization Design and Implementation

Following Figure 4.5, this subsection describes the details of the newly implemented version of the Onion approach. The central part of the user interface is dedicated to the Onion hypergraph representation. The panel on the left provides controls for various interactions and representation parameters. The import and export functions are available via the controls located on the top right. Furthermore, there is a floating button on the bottom right, which the user can click in order to view detailed data tables. The users can edit the underlying hypergraph data and filter various visible elements of the Onion view using these interactive options. The particular views and controls will be described in detail in the following subsections.

4.4.1 Data Import

As discussed above, our implementation allows data import in GraphML file. While the usual scenario is for loading a single file, the user can also load two GraphML

[1]URL: `https://insights.stackoverflow.com/survey/2019` (accessed: August 22, 2021)


Edge Bundling Threshold

Edge Bundling Intensity

- [x] Show Tooltip
- [x] Show Centric Line
- [x] Show Centric Nodes
- [x] Show Centric Hyperedges
- [x] Show Endpoints
- [x] Show Arc Hyperedges

Selected Options

- [✓] Nodes
- [✓] Endpoints
- [✓] Arcs
- [✓] Centric Hyperedges

Display Mode

- [ ] Sorting Mode
- [ ] Related Mode
- [ ] Cardinality Mode
- [ ] Comparison Mode

FileName: ggg.graphml Nodes: 19 Hyperedges: 30

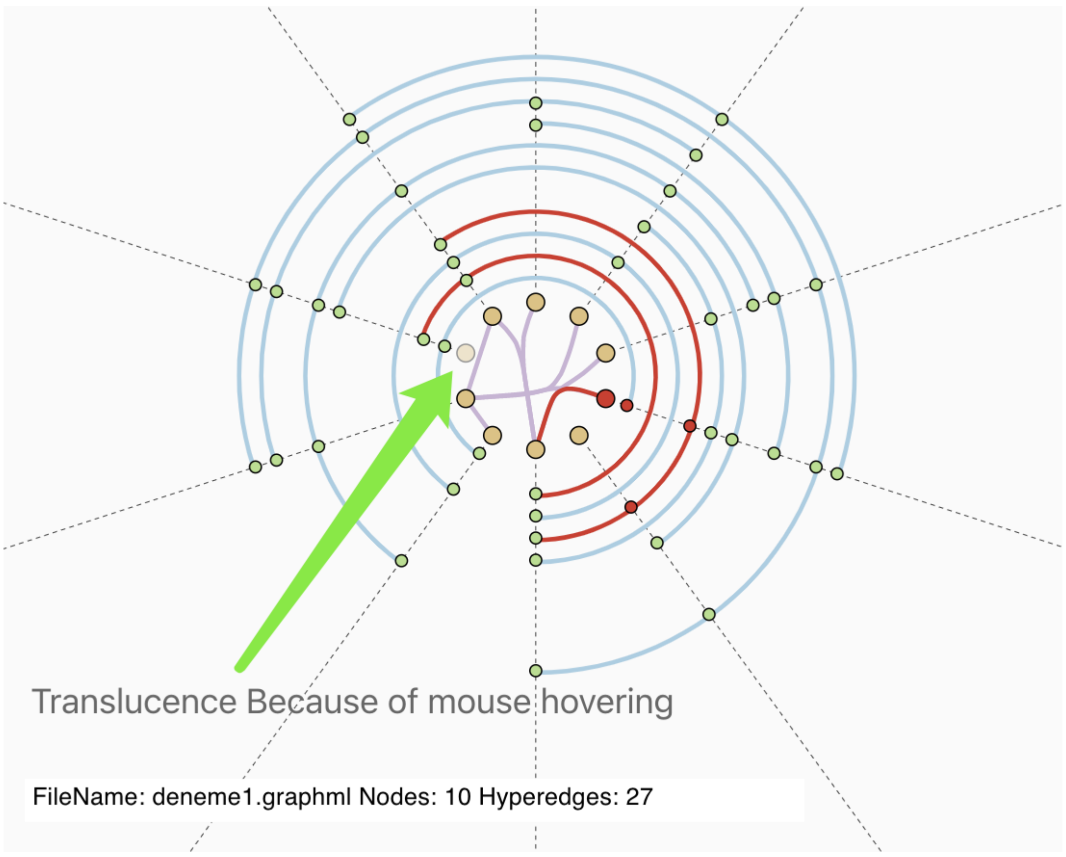
Figure 4.5: *A screenshot of the web-based Onion tool. The new Onion tool with a single hypergraph loaded. The main view is located in the center and the right part of the interface, and a panel with various controls is located on the left. Further controls are available on the top right and bottom right. According to the current view and the legend, we know that the file's name of the hypergraph is "ggg.graphml", the number of nodes is 19, and the number of hyperedges is 20.*

files to the Onion tool by pressing the import button displayed in Figure 4.6. To address the research question **RQ2**, we have implemented the comparison functionality in the Onion tool, and this is the reason why importing of two GraphML files simultaneously is supported.

Figure 4.6: *The file import interface.*

4.4.2 Main Hypergraph Representation

Onion representation view is the core part of the Onion tool. It transforms the GraphML into a radial layout visualization. As displayed in Figure 4.7, the hypergraph nodes are equally placed on a virtual circle as the yellow dots. In this implementation, the position of the nodes is random. As discussed in Section 4.1.1, only the hyperedges with a cardinality of at least three are represented as blue arcs surrounding the circle, and the inner area displays the binary/ordinary hyperedges as the purple curves that connect pairs of nodes. Considering the display space efficiency, we believe presenting the hyperedges in which the cardinality is less than two spawns unnecessary space occupied. Because of the above reason, we decide not to illustrate the hyperedges with a cardinality lower than two (however, the user has access to such data via the data editing dialog discussed below). The green dots of the blue arcs represent the endpoints of the hyperedge and indicate which nodes are part of the hyperedge. Hence, the user can follow the green dots towards the core layer to identify the hyperedge nodes (i.e., the yellow dots). Besides, we add the dotted lines to assist the user in locating the nodes and the endpoints.



FileName: deneme1.graphml Nodes: 10 Hyperedges: 27

Figure 4.7: *The main view of the new Onion tool with a single hypergraph loaded. According to the current view and the legend, we know that the file's name of the hypergraph is "deneme1.graphml", the number of nodes is 10, and the number of hyperedges is 27.*

In the bottom left corner of the main view, a legend shows the basic details of the current hypergraph model. The legend displays the file name of the hypergraph and the number of the hypergraph's nodes and hyperedges. Besides, selecting and highlight are the primary functions to distinguish the item which the user selected. The element itself is marked with translucence to distinguish it from the others by mouse hovering. The user can highlight the arcs, nodes, and endpoints in red by pressing the left mouse button. Moreover, the user can drag the mouse during the highlight mode for selecting multiple objects. To remove the highlight of the elements, the user needs to press the left mouse button again to cancel the highlight. The Onion diagram allows dragging and zooming events to present more details of the hypergraph to the user. If the user clicks and drags on the hypergraph model, it moves.

4.4.3 Tooltip

Tooltip is a graphical interface element that hovers over a screen object, and a text box displays the element's information nearby the element. Moreover, it is displayed as long as the mouse pointer is hovering the element. In the Onion tool, when the tooltip function is activated, the tooltips display a text label to identify the object. Except for some same features, such as file name and type, the tooltip shows different features depending on the objects. The explanation of each feature is following;

- *File Name*: The name of the file where the element belongs with.
- *Type*: The type of the element—a node, hyperedge, or endpoint.
- *Difference*: It is a unique feature only used when two hypergraphs were comparing with each other. The default value is false, and when two hypergraphs have a similar element, the value of the difference turns true.
- *Description*: A feature that provides additional information about the element to the user, if available in the source GraphML file, for instance.
- *ID*: A unique symbol of node and hyperedge to identify the element. The system assigns the ID to the hyperedge automatically. Besides, the node uses its own ID.
- *Cardinality*: A unique feature uses in the hyperedges. It provides a clear view for the user with the number of nodes connections the hyperedge has.
- *Name*: A unique feature uses in the endpoints. The name of the endpoint depends on which node is the endpoint projecting.
- *Path*: A unique auxiliary feature of the hyperedges that presents which nodes the hyperedge is connecting.

4.4.4 Dual Hypergraph Representation

To answer the research question **RQ2** of this project, we have implemented a function of the Onion tool to import two hypergraphs and compare them with each other. Inspired by the mirror image metaphor, following Figure 4.8, two hypergraphs are laid out as reflections of each other, but are reversed in the direction perpendicular to the mirror surface. When two hypergraphs are placed as a mirror image, we believe that the user can efficiently detect variations among two hypergraph models. This approach can be compared to a variation of the *juxtaposition* strategy for supporting comparison, as discussed by Gleicher et al. [26], albeit requiring a customization of the layout in contrast to naïve juxtaposition. Besides, we have implemented the comparison visual mode to assist the user in data analyzing, as mentioned in Section 4.4.8. However, we have faced a rather severe issue: we cannot find an adequate solution to identify and track an element *after* two hypergraphs are imported. Since we cannot accurately distinguish the element which the user selects, we disable the data editing function when two hypergraphs are loaded simultaneously.

4.4.5 Control Panel

If the Onion diagram is the skin of the Onion tool (as it is the main view that the user is presented with immediately), then the control panel is the skeleton of this tool. The control panel controls the encoding and behavior of the interactive Onion visualization. It has different custom options to interact with the view and help the user discover the structure of the input hypergraph and analyze it further. A number

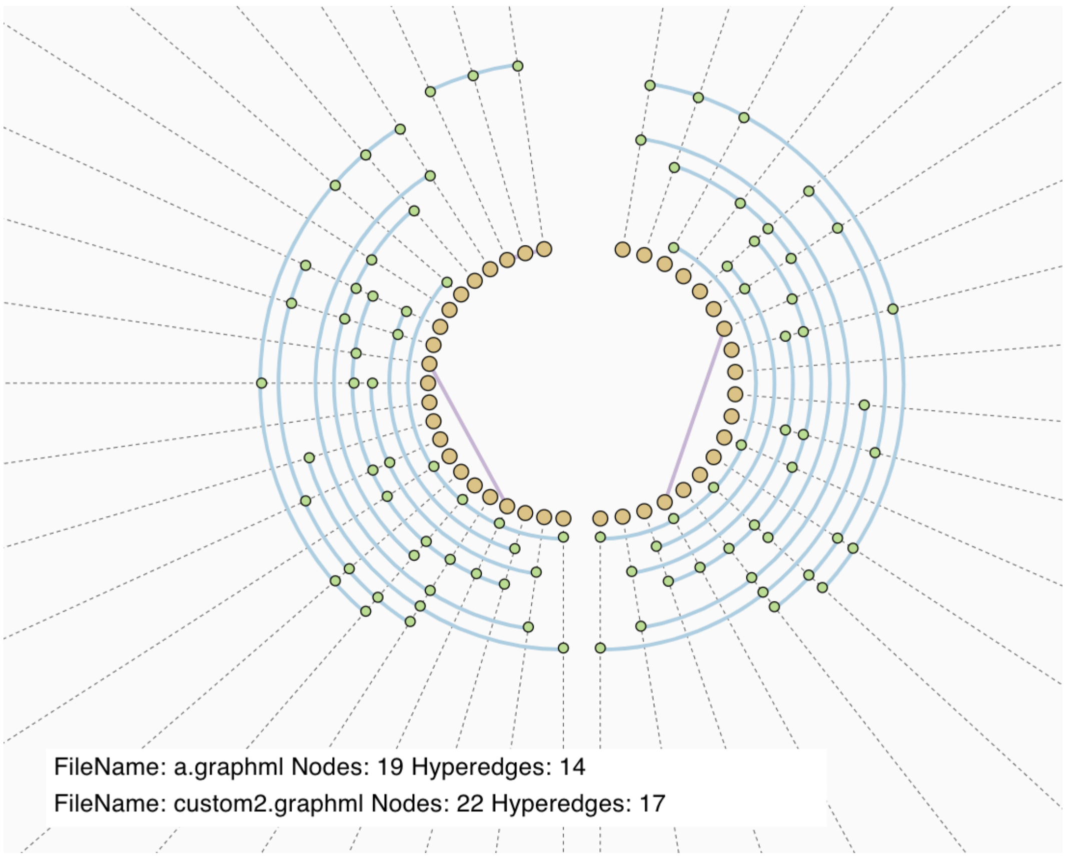
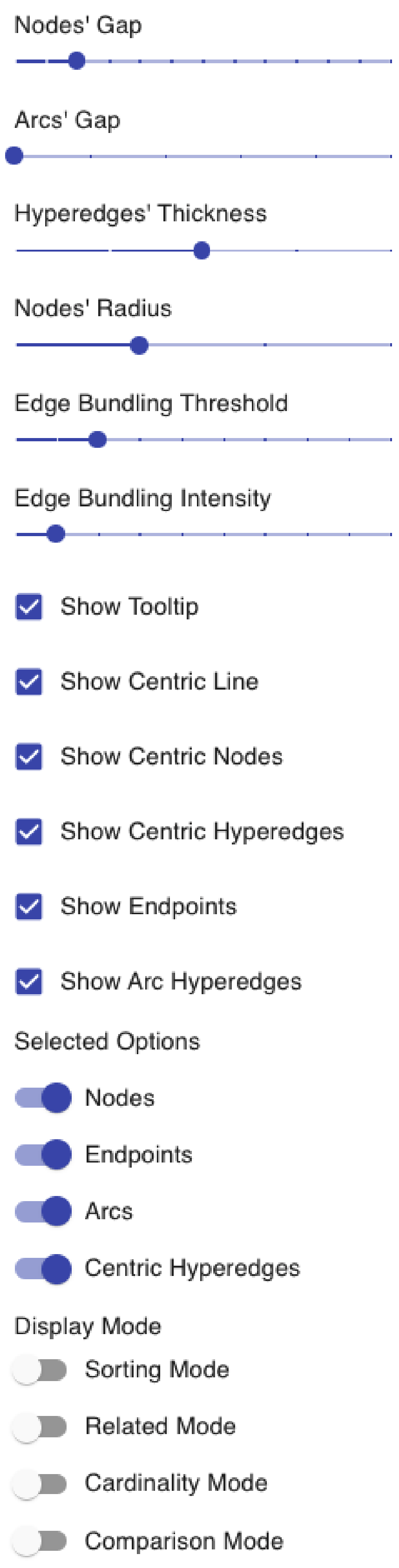

FileName: a.graphml Nodes: 19 Hyperedges: 14
FileName: custom2.graphml Nodes: 22 Hyperedges: 17

Figure 4.8: *The main view of the new Onion tool with two hypergraphs loaded. According to the current view and the legend, we know that "custom2.graphml" has 22 nodes and 17 hyperedges, and "a.graph.graphml" has 19 nodes and 14 hyperedges.*

of standard techniques are offered as in the previous Onion tool version, such as changing a set of layout parameters (distance between nodes and arcs, the thickness of arcs, or the radius of the nodes). Of course, it is also possible to hide, display, disable, and highlight the hypergraph's elements when the user wants to concentrate on particular ones. In the following, we concisely review the essential features of the control panel that facilitate the analysis process.

Firstly, since we add force-directed edge bundling to our Onion approach, we have *Edge Compatibility Score* and *Node-Link Step Size* sliders that providing the parameters of the force-directed edge bundling method to organize the adjacent relations between edges to increase/decrease the clutter and providing a different observed aspect in complex networks. We will provide more details in Section 4.4.7.

Additionally, we implement different visual modes for our visual project: *sorting mode*, *cardinality mode*, *comparison mode*, and *related mode* are the visual modes for aiding the user to export further information of the hypergraph model. Besides, the comparison mode only can be used when two hypergraphs are imported. Regrading the visual modes, we will present them in Section 4.4.8 with further details.


Nodes' Gap

Arcs' Gap

Hyperedges' Thickness

Nodes' Radius

Edge Bundling Threshold

Edge Bundling Intensity

✓ Show Tooltip

✓ Show Centric Line

✓ Show Centric Nodes

✓ Show Centric Hyperedges

✓ Show Endpoints

✓ Show Arc Hyperedges

Selected Options

● Nodes

● Endpoints

● Arcs

● Centric Hyperedges

Display Mode

○ Sorting Mode

○ Related Mode

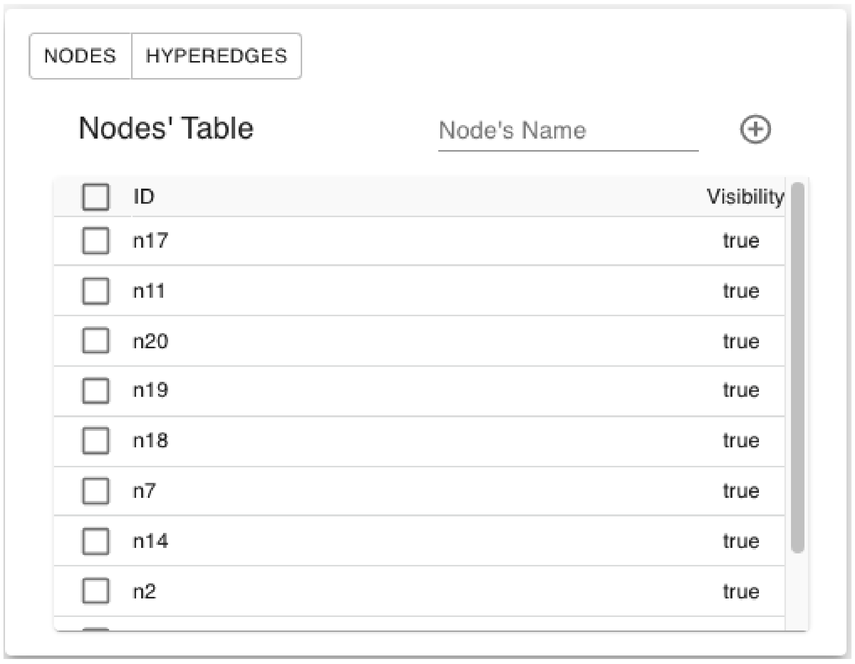
○ Cardinality Mode

○ Comparison Mode

Figure 4.9: *The user can change the Onion visualization by manipulating the parameters of the control panel. We provide more details in Section 4.4.5.*

4.4.6 Data Editing

To complete the requirements of the previous Onion future study mentioned in Section 1.1.4, we develop the data editing functionality to support creation, removal, and filtering/hiding the hypergraph elements for our Onion tool.



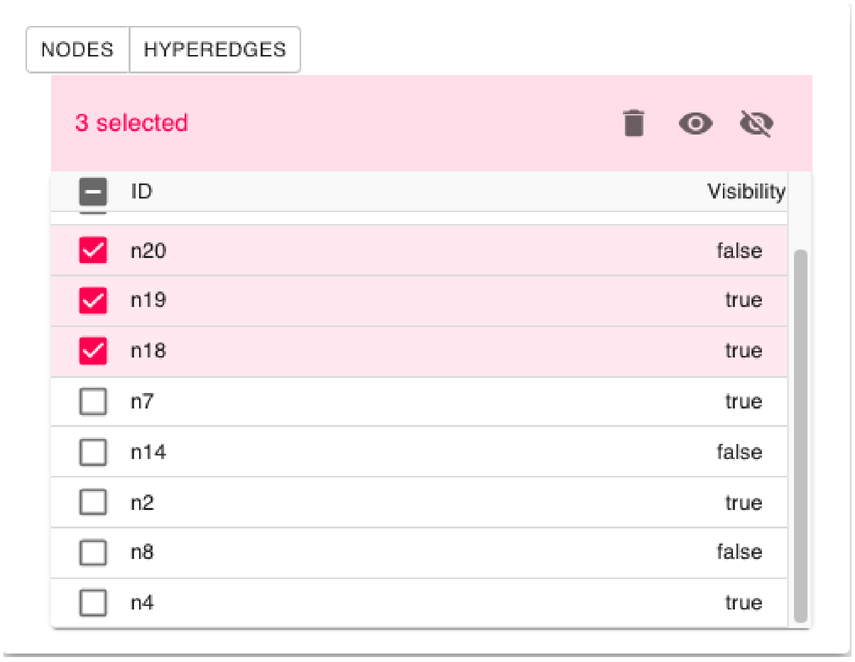
ID	Visibility
n17	true
n11	true
n20	true
n19	true
n18	true
n7	true
n14	true
n2	true

Figure 4.10: *The nodes table shows the IDs and the visible status of the nodes. Each node has its unique ID. If the visibility of the node is true, it means it is currently displayed in the main view, and vice versa.*

As Figure 4.10 shows, the user opens the nodes table by clicking the floating button on the right corner (the design for the hyperedges table is similar in this regard). We can see it has two columns (*ID* and *Visibility*) and a checkbox control element. There is a text input component on the top of the table. The user can add a new node to the hypergraph by entering the ID of the node. However, to ensure the uniqueness of the elements, the nodes are not allowed to have the same ID. Following Figure 4.11, the user can select the nodes by checkbox, and then the action selections will show on the top of the nodes table to remove, show, or hide the nodes. When the element has been hidden, the value of the elements' visibility feature will turn to "false", but the layout of the Onion diagram will not be affected. It is because we believe changing the layout of the Onion diagram may create confusion for the user. For removing the element, the user can press the bin icon to remove the nodes of the hypergraph.

As Figure 4.12 presents, the hyperedges table has four columns: *ID*, *Cardinality*, *Visibility*, and *Path*. Compared with the nodes table, the ID of the hyperedge is assigned by the system automatically. Besides, the hyperedges table follows a more complex scenario than the nodes table with regard to editing. To elaborate on this, the scenarios of the hyperedges data editing are the following:

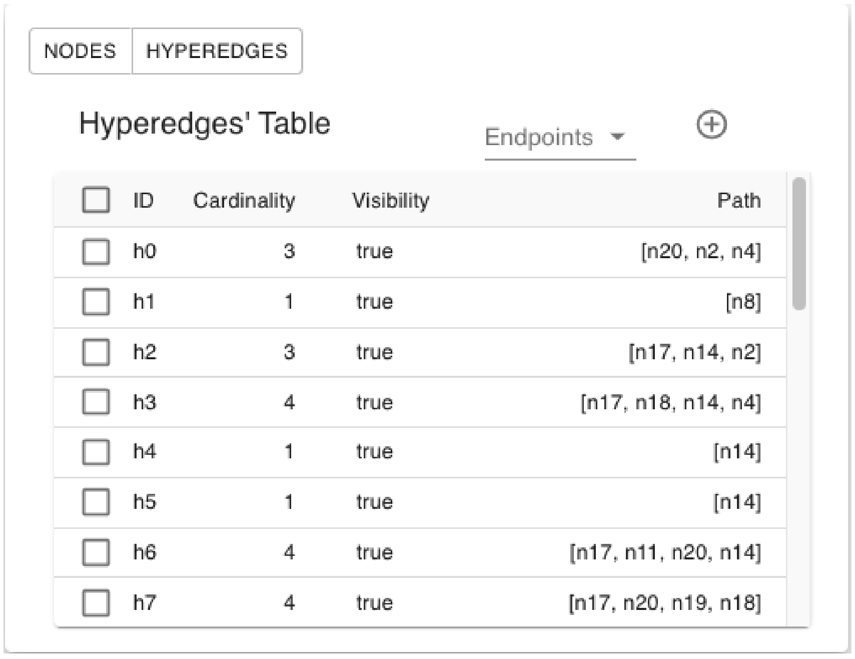
- Creating a new hyperedge with existing nodes
- Adding nodes to existing hyperedges



ID	Visibility
n20	false
n19	true
n18	true
n7	true
n14	false
n2	true
n8	false
n4	true

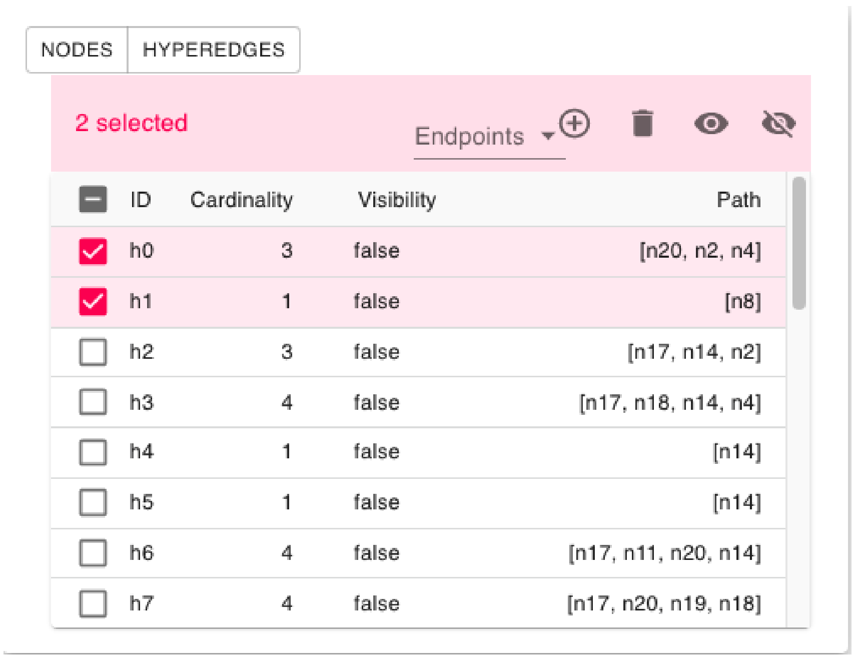
Figure 4.11: *Selection example. Here, three nodes'* n18, n19, *and* n20 *backgrounds turn red and have been selected in the nodes table.*

To support the above scenarios, we add a select component for listing and selecting all the existing nodes (see Figure 4.13). When creating a new hyperedge, we can select the node names in the select component and press the cross icon button. Moreover, we select the hyperedges after selecting the nodes in the select component to add new nodes into the existing hyperedges (see Figure 4.14).



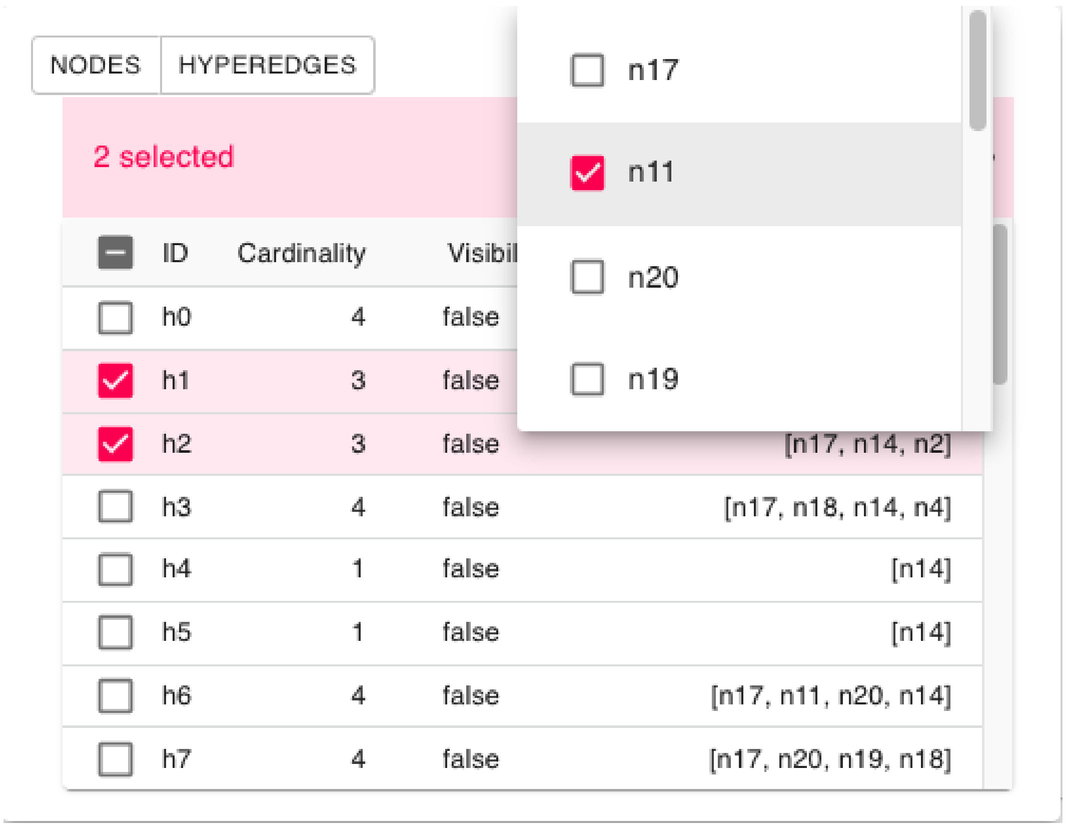
ID	Cardinality	Visibility	Path
h0	3	true	[n20, n2, n4]
h1	1	true	[n8]
h2	3	true	[n17, n14, n2]
h3	4	true	[n17, n18, n14, n4]
h4	1	true	[n14]
h5	1	true	[n14]
h6	4	true	[n17, n11, n20, n14]
h7	4	true	[n17, n20, n19, n18]

Figure 4.12: *Compared with the nodes' table, the hyperedges table shows more features. The system assigns IDs of the hypergraphs automatically. Cardinality provides the number of nodes connections the hyperedge has. Furthermore, the path shows the IDs of the nodes which are connected by the hyperedge.*



☐	ID	Cardinality	Visibility	Path
✓	h0	3	false	[n20, n2, n4]
✓	h1	1	false	[n8]
☐	h2	3	false	[n17, n14, n2]
☐	h3	4	false	[n17, n18, n14, n4]
☐	h4	1	false	[n14]
☐	h5	1	false	[n14]
☐	h6	4	false	[n17, n11, n20, n14]
☐	h7	4	false	[n17, n20, n19, n18]

Figure 4.13: *Selection example. Here, two hyperedges* h0 *and* h1 *have been selected in the hyperedges table.*



☐	ID	Cardinality	Visibil	
☐	h0	4	false	
✓	h1	3	false	
✓	h2	3	false	[n17, n14, n2]
☐	h3	4	false	[n17, n18, n14, n4]
☐	h4	1	false	[n14]
☐	h5	1	false	[n14]
☐	h6	4	false	[n17, n11, n20, n14]
☐	h7	4	false	[n17, n20, n19, n18]

Figure 4.14: *Data editing example. Here, the node* n11 *has been selected and is currently being added to the existing hyperedges* h1 *and* h2.

4.4.7 Edge Bundling

Edge bundling is a method that combines geometrically close edges into bundles, which use much less screen space [78]. To find a solution to answer our research

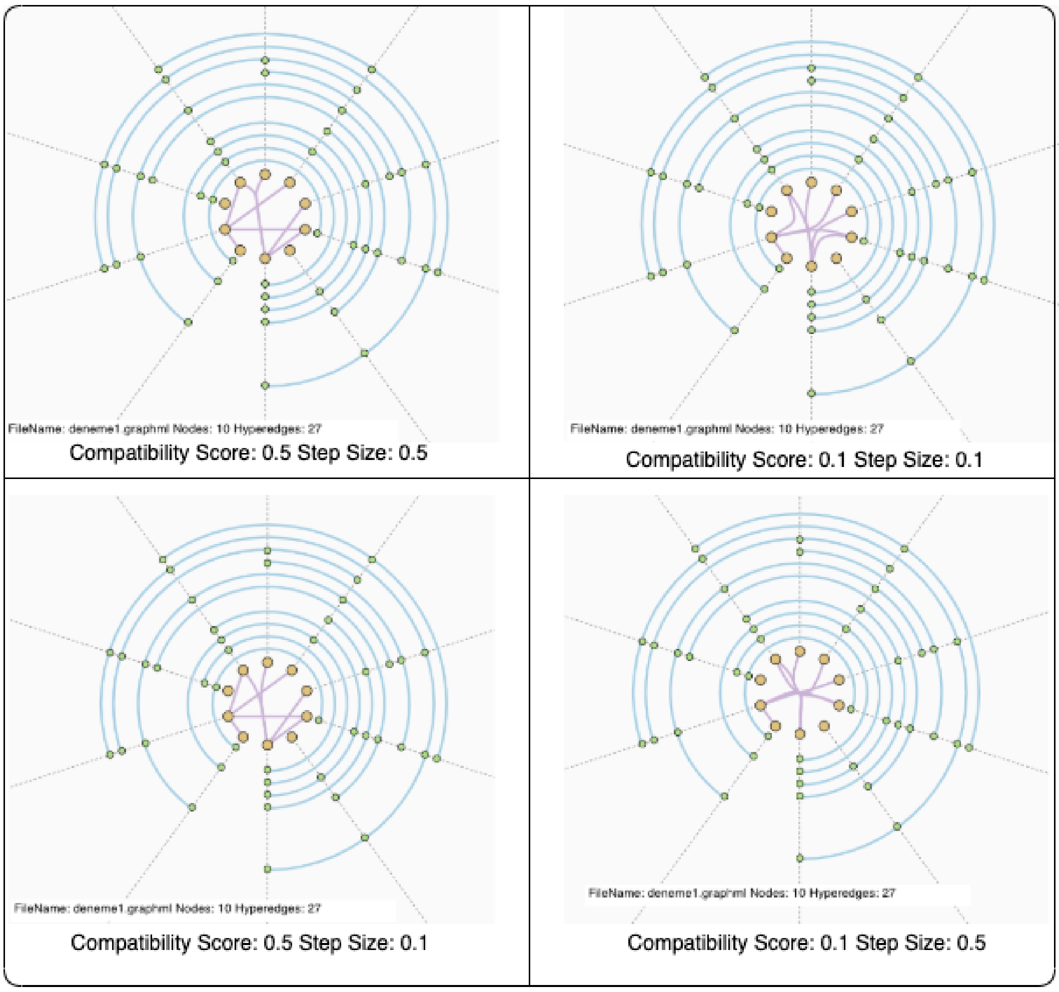

FileName: deneme1.graphml Nodes: 10 Hyperedges: 27
Compatibility Score: 0.5 Step Size: 0.5
FileName: deneme1.graphml Nodes: 10 Hyperedges: 27
Compatibility Score: 0.1 Step Size: 0.1
FileName: deneme1.graphml Nodes: 10 Hyperedges: 27
Compatibility Score: 0.5 Step Size: 0.1
FileName: deneme1.graphml Nodes: 10 Hyperedges: 27
Compatibility Score: 0.1 Step Size: 0.5

Figure 4.15: *Under the same hypergraph data, the results of force-directed edge bundling are different with various paraments. When the compatibility score = 0.1 and step size is 0.5, the hyperedges are bundled together and hard to distinguish.*

question **RQ1**, we apply edge bundling method inside the inner area of the main Onion view. There are many options available for us with regard to edge bundling approaches, as we discussed in Section 3.4. One of the standard methods is the hierarchical edge bundling. Hierarchical edge bundling is a flexible and generic method used in conjunction with existing tree visualization techniques [65]. However, in this project, we only consider undirected/unordered hyperedges. One of the undirected hyperedge characteristics is that each hyperedge does not have a strong correlation with the other. Moreover, it is not evident which hierarchical clustering scheme or spanning-tree generation method would suit such a task. To apply the hierarchical edge bundling to our project, we need to pay a high price in inventing an algorithm to alter the hypergraph structure, so a different solution is required.

As we mentioned in Section 3.4, force-directed edge bundling behavior is easy to understand because of the straightforward physics model. Force-directed edge bundling uses an intuitive, self-organizing approach to bundling by modeling edges as flexible springs that can attract each other without generating a control mesh or a hierarchy data structure [67]. Thus, we follow this approach in our implementation.

Force-directed edge bundling has two parameters that essentially harmonize the algorithm to produce functional diagrams for graphs. *Compatibility score* considers the compatibility between the parting edges. The range of compatibility scores should be between 0 and 1. The most crucial argument of force-directed edge bundling is the *step size*, which is considered when moving the subdivision points after forces have been computed. The step size calculation method is also affected by the scale of the graph, i.e., the number of edges and nodes contained. A low step size value will produce node-link-like graphs, while too high values will over-distort edges. An example in Figure 4.15 demonstrates the results with different parameters values.

4.4.8 Visual Modes

Visual modes (see Figure 4.16) provide further visual assistance to help the user to explore the data. There are four visual modes in the Onion tool: *sorting mode*, *cardinality mode*, *related mode*, and *comparison mode*.

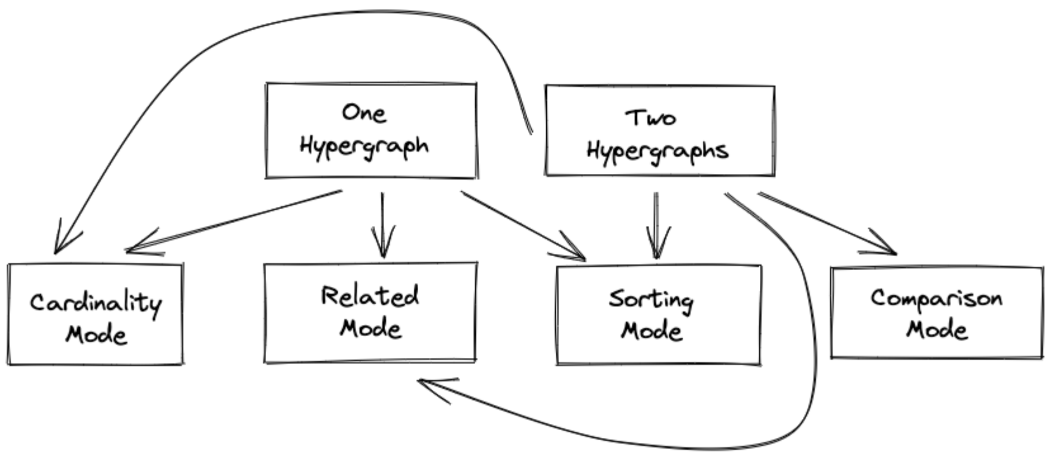


Figure 4.16: *The supported visual modes of the new Onion tool. The Onion tool does not support the comparison mode when two hypergraphs are imported.*

Sorting mode arranges the hyperedges with a cardinality that is larger than two from low to high. It means that an hyperedge with low cardinality is placed closer to the inner layer (see Figure 4.17). On the other hand, a hyperedge with high cardinality is laid out closer to the outer layer (Figure 4.18).

Displayed in Figure 4.19, cardinality mode provides a different aspect for the user to observe the hypergraph. Cardinality mode divides the hyperedge into six groups based on the hyperedges' cardinality and gives each group a color. The user can discover hyperedges' cardinality with an approximate number by the legend displayed in the left top corner of the main view panel.

To address the future work suggestions of the previous study of Onion, we have implemented the related mode to help the user quickly determine all hyperedges that share a specific node (or set of nodes). The related hyperedges/nodes are highlighted only if a specific hyperedge or node is hovered over by the mouse pointer. Both nodes and hyperedges can be alternatively selected using the lists in the control panel, as shown in Figure 4.20.

Our Onion tool can import two hypergraphs. For helping the user investigate more details of the data, we have implemented the comparison mode. From Figure 4.21, when the comparison mode turns on, the system will compare the element IDs between two hypergraphs to find the elements they match. If such elements are detected, the value of "difference" is equal to true and the color of the respective visual elements is changed to green. Otherwise, the value of "difference" is set to false and the color of visual elements is set to brown. Besides, the elements with duplicate IDs become translucent when the mouse pointer is hovering over them.

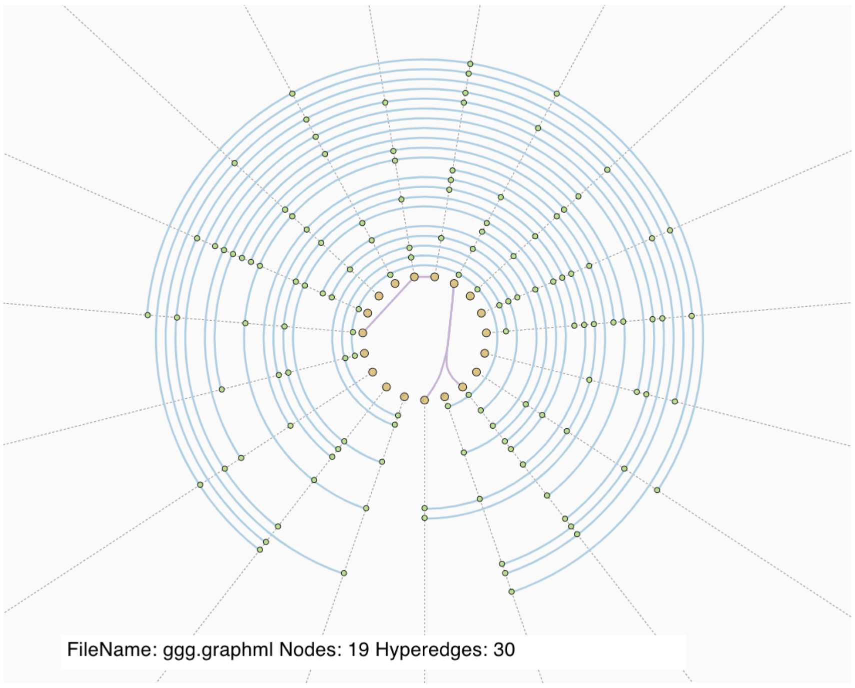


Figure 4.17: *A hypergraph before the sorting mode is turned on. All hyperedges with a cardinality of at least three are placed randomly.*

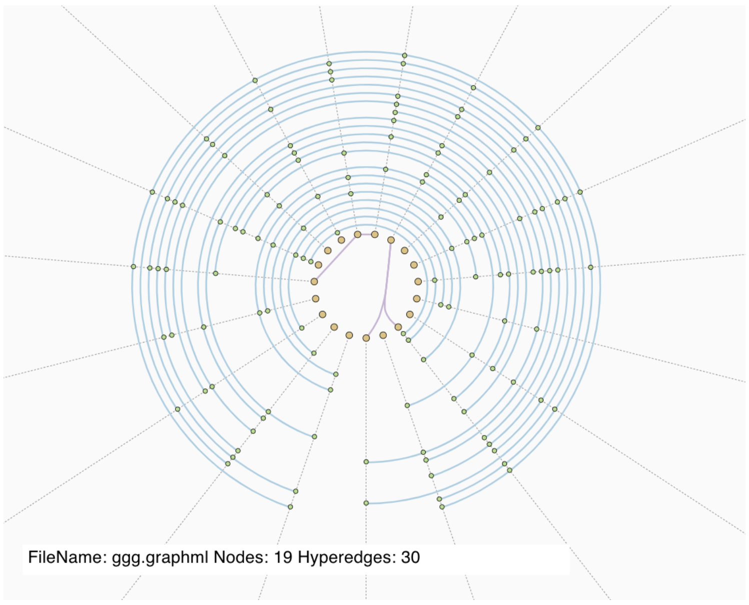
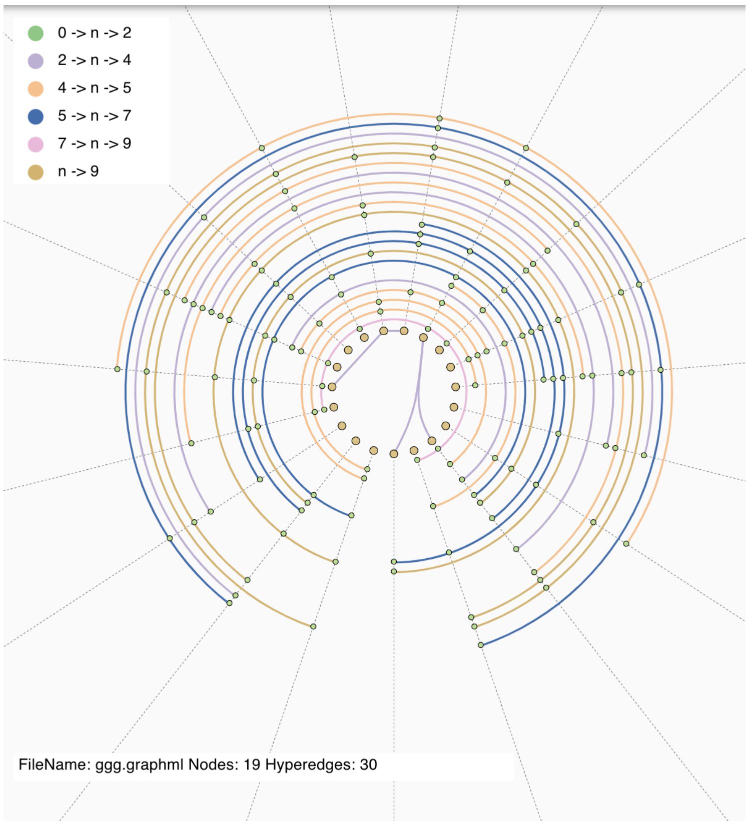


Figure 4.18: *A hypergraph after the sorting mode is turned on. All hyperedges with a cardinality of at least three are placed from lower to higher layer according to the cardinality.*



FileName: ggg.graphml Nodes: 19 Hyperedges: 30

Figure 4.19: *The user can divide the hyperedges into six groups based on the cardinality value when the cardinality mode is turned on.*

4.4.9 Data Export

As well as the previous Onion approach, our implementation provides a method for the user to save the data in different formats, namely, PNG, JPEG, or GraphML. However, we cannot provide a solution to transform two hypergraphs into one GraphML file without significantly changing the focus and scope of this project, and thus our approach does not support saving the data as GraphML when two hypergraphs are imported.

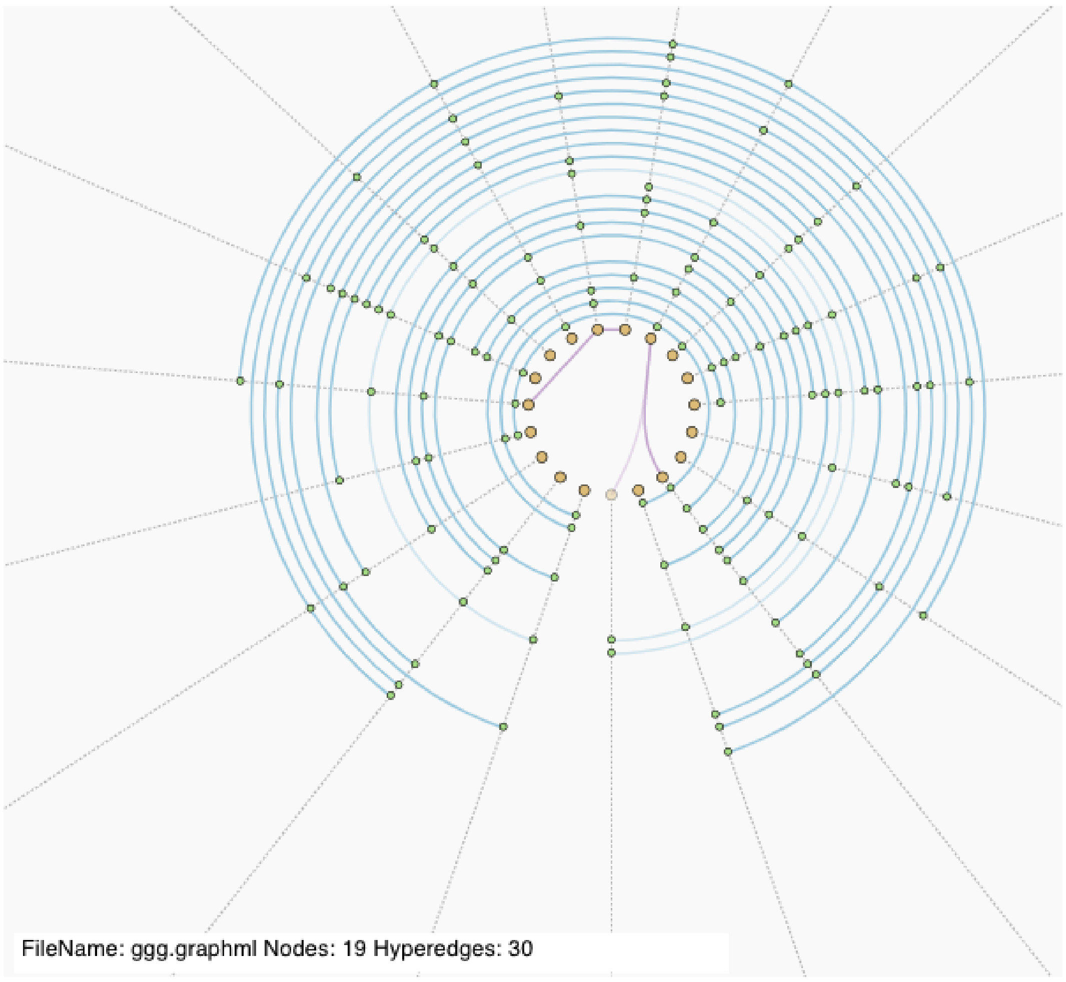
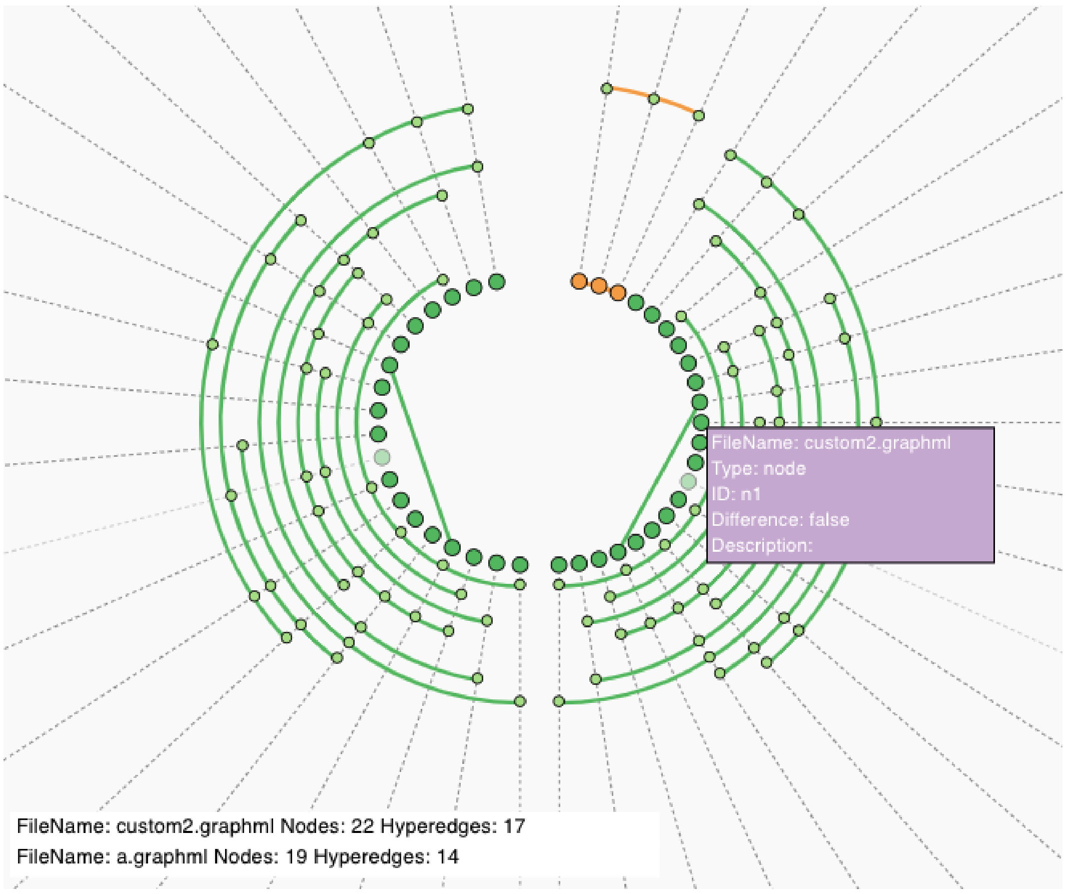


Figure 4.20: *The related hyperedges/nodes are highlighted when a specific hyperedge or node is hovered upon by the mouse pointer while the related mode is turned on.*



FileName: custom2.graphml Nodes: 22 Hyperedges: 17
FileName: a.graphml Nodes: 19 Hyperedges: 14

Figure 4.21: *The comparison mode helps the user detect the difference between two hypergraphs in a quick way. The elements which have identical IDs change opacity to translucent when the mouse pointer is hovering over them.*

5 Evaluation

Evaluation is an important part of the design and implementation process in human-computer interaction [34] and visualization [35,37,38], which helps the designers to validate their proposed solutions and estimate its usability and further characteristics. This project will use several methods to determine the usability of the proposed web-based Onion approach, such as (1) case studies that involve real-world data to handle several hypergraphs that share some of the nodes and edges, and (2) a user study that mainly focuses on qualitative user feedback and questionnaires that aim to analyze how the conceived visualization fits the requirements. Then we concentrate on analyzing the results. Afterwards, we discuss the scalability of the implementation in several aspects. Finally, we discuss the pros and cons of our implementation in comparison with other related approaches.

5.1 Case Studies

A *case study* is a research methodology that is commonly seen in information visualization investigation [37, 38]. It can be described in this context as a data analysis study involving a visualization approach in which the researchers (potentially together with experts from other disciplines or domains) examine real-world data in-depth using the respective interactive visualization technique or tool.

This part of the project investigates our Onion approach using several actual data sets collected from the real world and provided by the ISOVIS group. Following our research targets mentioned in Section 1.4, we conduct two case studies with different real-world data sets. In the first study, we concentrate on the intersections of the bus routes in Växjö, Sweden in 2012 and 2020. Then, in the second study, we investigate two hypergraphs based on the publication data fetched from the Linnaeus University bibliographical database DiVA. All of these data sets were provided by the supervisor of this project, Dr. Kostiantyn Kucher.

5.1.1 Case Study: Bus Route Intersections in Växjö, Sweden in 2012 and 2020

The data is based on the bus route maps for Växjö, Sweden as provided by Länstrafiken Kronoberg in 2012[2] and 2020[3]. It uses the information about each bus line's stops as nodes and intersections between the lines (i.e., shared bus stops) as (hyper-)edges. The data has been transformed into GraphML and is analyzed in the Onion tool.

We firstly take a look at 2012, as demonstrated in Figure 5.1. This data set contains 8 nodes and 39 hyperedges. Following the sorting mode, according to the endpoints number, we quickly find out that Lines 2 and 6 have the largest number of transfer stations in this data set. This is an expected outcome, since these two bus lines in Växjö are designed as almost exact mirror copies running in opposite directions; however, discovering this insight in Onion helps us verify that the tool parses and represents the respective data faithfully. Next, the second most intersections are concentrated on Lines 7 and 8 (which shared large parts of their routes with

[2]URL: `http://assets.wm3.se/sites/16/media_files/175/2VFWkE5y4OV_ROzKk6x2Hg/original_linjeschema_2012.pdf` (accessed: August 20, 2021)

[3]URL: `http://static.wm3.se/sites/268/media/466009_Linjekarta_V%C3%A4xj%C3%B6stadstrafik_31_augusti2020_webb.pdf` (accessed: August 20, 2021)

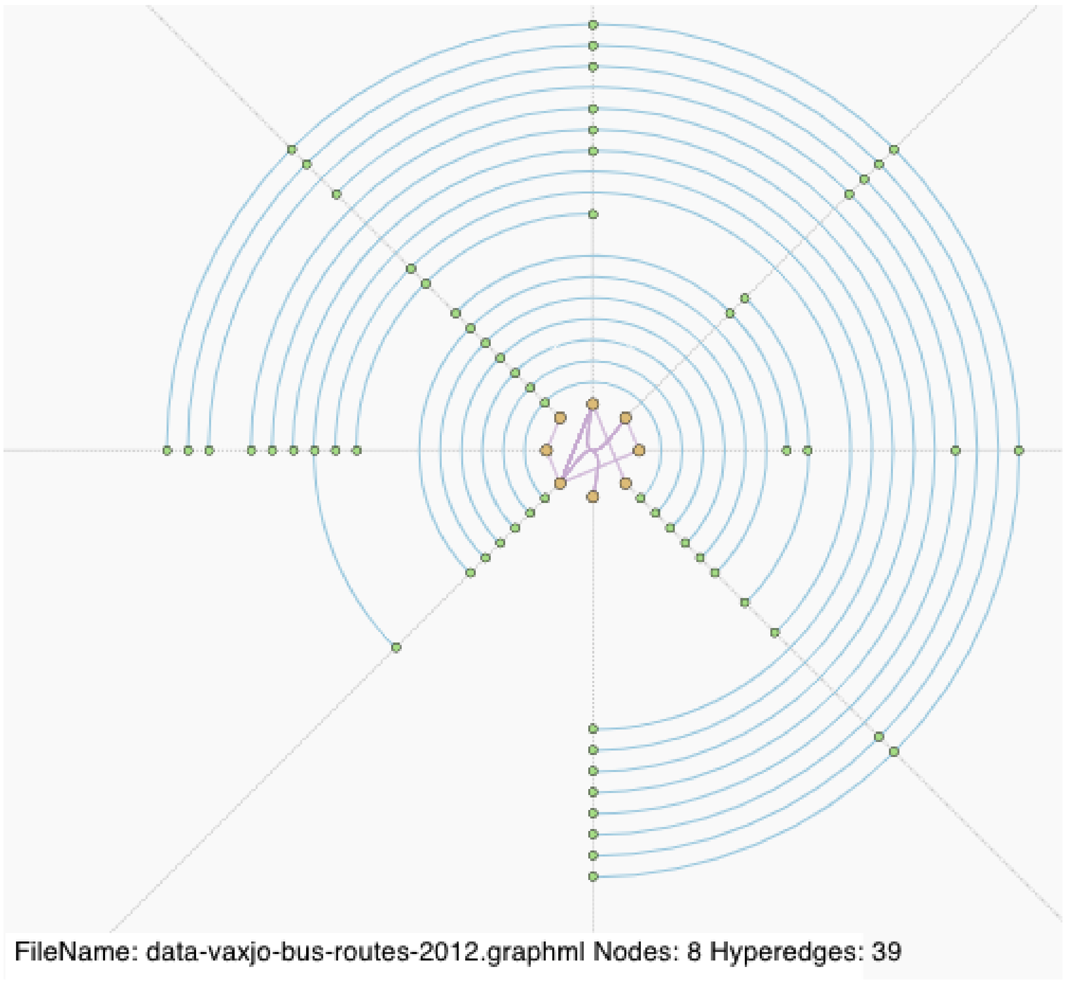
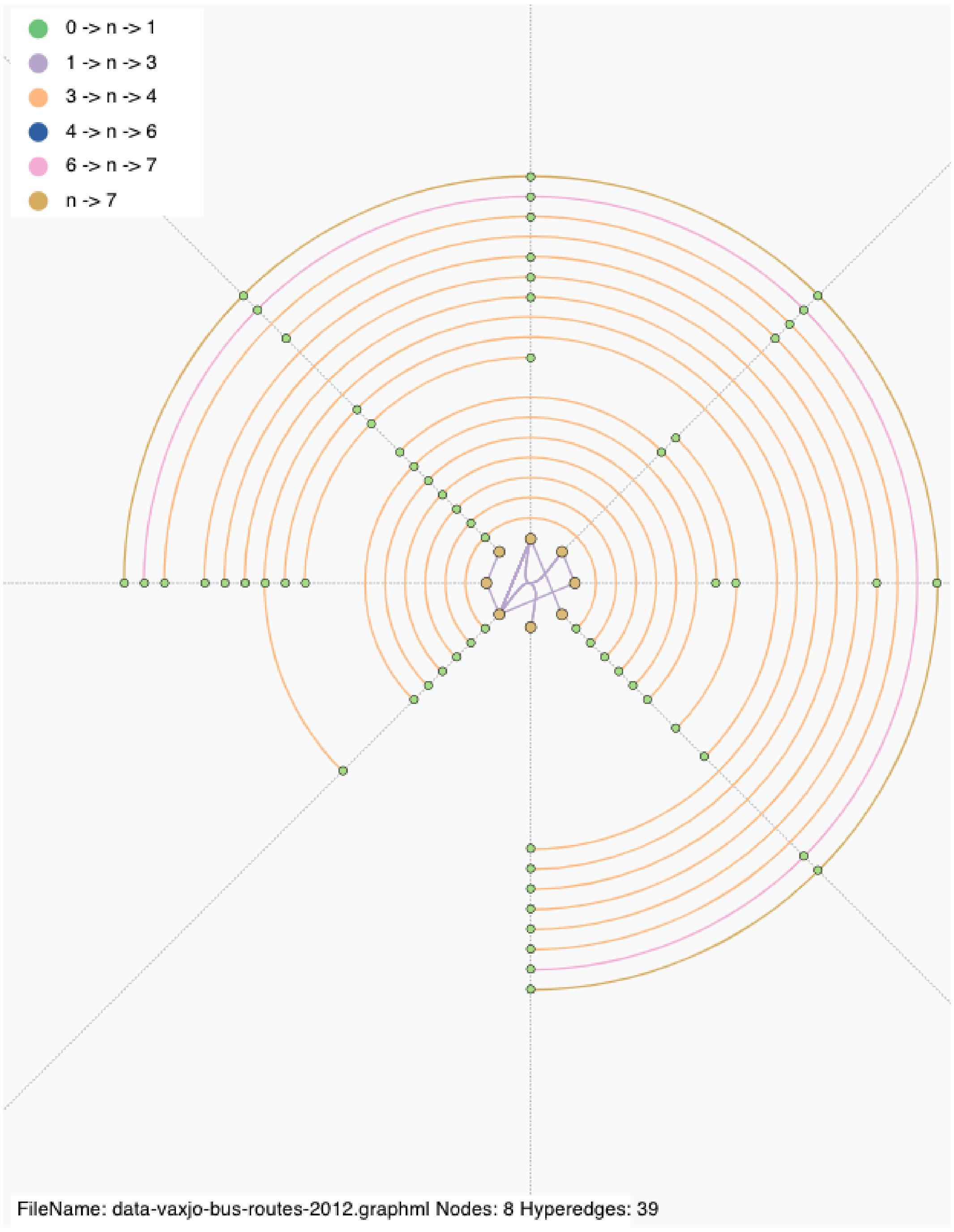

FileName: data-vaxjo-bus-routes-2012.graphml Nodes: 8 Hyperedges: 39

Figure 5.1: *Exploration of the 2012 Växjö bus routes data with the Onion tool. According to the number of endpoints, we quickly find out that Lines 2 and 6 have the largest number of transfer stations here.*

other bus lines). Under the sorting mode, we can see that Lines 2, 6, and 8 share six transfer stations. Besides, Lines 1, 5, and 7 have three shared transfer stations. Then, with the cardinality mode (Figure 5.2), we notice that most lines have 3 to 4 shared transfer stations, and the Resecentrum station has the largest number of bus lines passing through (this finding also makes sense, as Resecentrum was the main hub of town and regional bus traffic as well as the train station in Växjö in 2012).

Next, we concentrate on the 2020 bus routes data set, as presented in Figure 5.3. It includes 11 nodes and 39 hyperedges. Based on the layout of the Onion, we can discover that Lines 2 and 6 are the main routes that connect most of the bus lines. Lines 1, 3, and 5 have three overlapping bus stops. Besides, Lines 2, 6, and 8 also have three shared bus stops. Line 15—which connects the town center with one of the industrial areas—does not have a strong connection with other bus stations; if people want to take Line 15, they can only go to Stortorget station. Using the cardinality mode (Figure 5.4), we can find out that the Stortorget bus station has the largest cardinality number compared with the other bus stops (i.e., hyperedges), which is true, as that station has acted as the primary bus transfer hub as of 2020.

Finally, we import both the 2012 and 2020 bus route data sets to Onion and investigate the difference under the comparison mode, as displayed in Figure 5.5. With the help of comparison mode, we can quickly notice that the number of trans-



0 -> n -> 1
1 -> n -> 3
3 -> n -> 4
4 -> n -> 6
6 -> n -> 7
n -> 7
FileName: data-vaxjo-bus-routes-2012.graphml Nodes: 8 Hyperedges: 39

Figure 5.2: *Using the cardinality mode with the 2012 Växjö bus routes data. Here, we can find out that most lines have 3 to 4 shared transfer stations, and the Resecentrum station has the largest number of passing bus lines.*

fer stations has not changed, but three new bus lines (indicated with orange-colored nodes) have emerged in the newer data set: Lines 9, 12, and 15. Besides, by the mirror image layout assisting, we can confirm that the paths of Lines 2, 6, and 8 have not changed (as represented with some of the green-colored hyperedges). Four of the intersection stops of Line 7 have been abolished — this finding is related to the fact that historically the routes of Lines 3 and 7 went through a major revision in 2013.

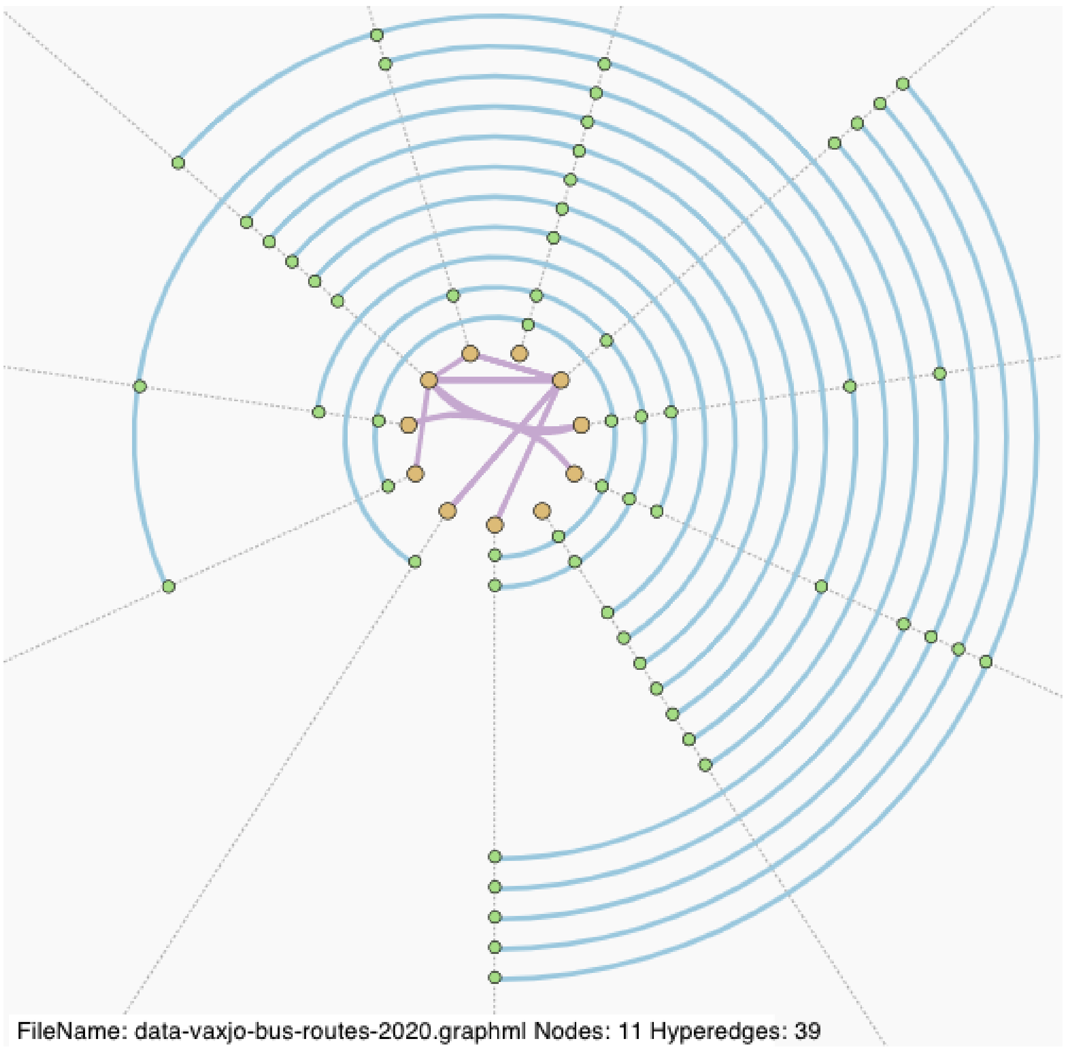

FileName: data-vaxjo-bus-routes-2020.graphml Nodes: 11 Hyperedges: 39

Figure 5.3: *Exploration of the 2020 Växjö bus routes data with the Onion tool. According to the number of endpoints, we quickly find out that Lines 2 and 6 have the largest number of transfer stations in this data, similar to the older data set.*

While we have been able to investigate these real-world data sets and confirm a number of our expectations (based on the respective real-world knowledge), we have also faced several challenges, primarily during the Onion central area investigation. The first problem is edge bundling. Since some of the hyperedges are merged, it is hard to recognize the connection between the nodes intuitively. Secondly, some of the nodes overlay the middle hyperedges. Finally, while we were able to associate the nodes and hyperedges with the respective labels and descriptions as part of the interactive exploration process, the labels are missing from the figures provided in this case study—the alternative would involve further severe cluttering, though.

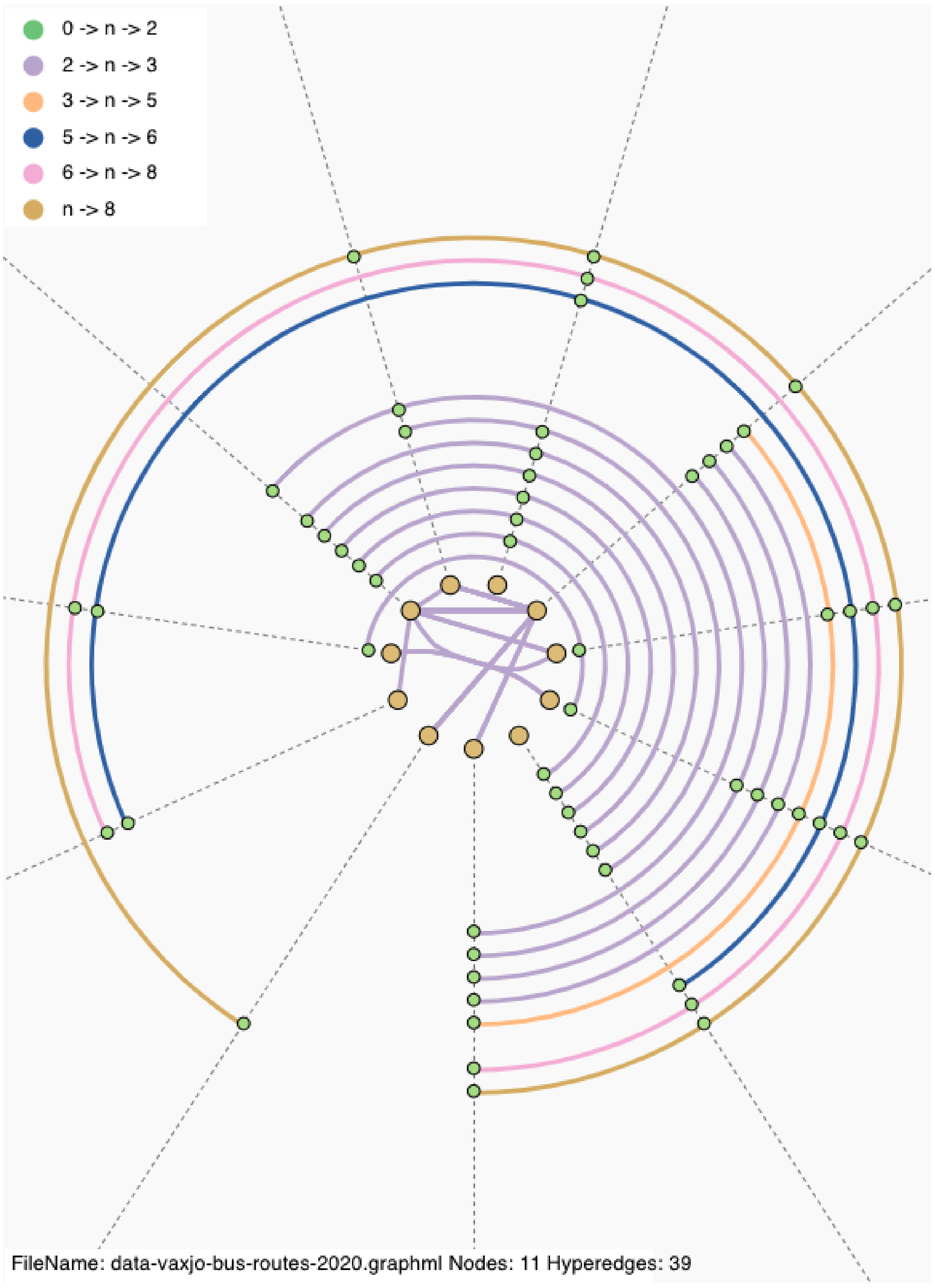

0 -> n -> 2
2 -> n -> 3
3 -> n -> 5
5 -> n -> 6
6 -> n -> 8
n -> 8
FileName: data-vaxjo-bus-routes-2020.graphml Nodes: 11 Hyperedges: 39

Figure 5.4: *Using the cardinality mode with the 2020 Växjö bus routes data. Here, the Stortorget station has the largest cardinality number compared with other hyperedges, and thus we can identify that the Stortorget station is the primary transfer bus station according to the respective year's data.*

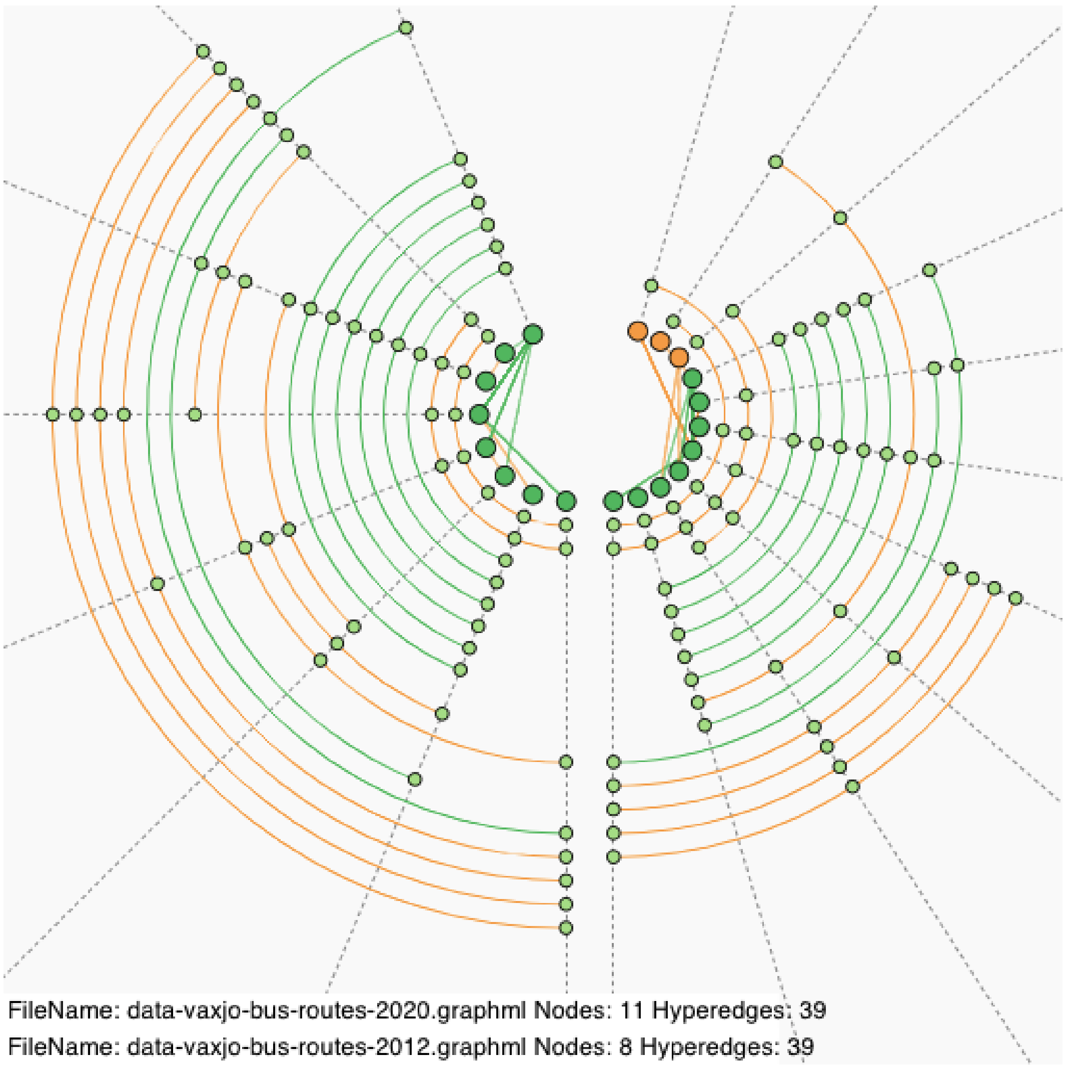

FileName: data-vaxjo-bus-routes-2020.graphml Nodes: 11 Hyperedges: 39
FileName: data-vaxjo-bus-routes-2012.graphml Nodes: 8 Hyperedges: 39

Figure 5.5: *Comparison between Växjö bus routes for 2012 and 2020 using the comparison mode in the Onion tool. In the 2020 data set, three new bus lines have emerged Växjö compared to 2012 (denoted by orange nodes in the inner layer).*

5.1.2 Case Study: Departmental Publication Data for 2019 and 2020 from DiVA

The data used for the user study is based on publication data fetched from the Linnaeus University bibliographical database DiVA[4]. It includes two data sets with peer-reviewed publications authored by the Department of Computer Science and Media Technology staff at LNU during 2019 and 2020, respectively. The data sets were extracted from DiVA and converted to GraphML files. Each of these GraphML files includes nodes representing the faculty members (using internal IDs, such as “kokuaa” for “Kostiantyn Kucher”, when available) and hyperedges corresponding to the publications co-authored by these faculty members through the respective year. However, former LNU staff or collaborators from other departments or faculties within LNU are also included (due to the peculiarities of the data extraction process). Thus, the two resulting files include a number of common nodes (same staff members) and some hyperedges with the same endpoints (indicating collaboration between the same groups of paper co-authors).

We start our analysis with the 2019 data set, which includes 62 nodes and 74 hyperedges, as demonstrated in Figure 5.6. When using both the sorting mode and the cardinality mode, we get a good overview of the structure of the hypergraph as well as the most prominent nodes and hyperedges. Compared to the previous case study, we can also witness better outcomes of the edge bundling algorithm in the central area, which allows us to quickly grasp the main two-node interactions with less edge clutter (thus, we can surmise that the effectiveness of edge bundling in our tool depends on the size of the hypergraph and the available central area space, respectively). With regard to the data, we quickly find out the most frequent cardinality of the hyperedges is 2–4; however, some hyperedges involve 5 nodes (i.e., paper co-authors), and there is even a hyperedge with 6 nodes, which corresponds to the 2019 publication by Alissandrakis et al. [79]. By using the related mode, we can also notice the publications corresponding to a particular author, while hovering over the respective node. By using Onion for this data set, we are also able to focus not just on the number of publications of a particular author, but rather on the co-authorship pattern, e.g., whether an author preferred to publish alone or with 1–2 collaborators, or a relatively large group of co-authors.

Next, we switch to the analysis of the 2020 data set, which is presented in Figure 5.7. It includes 62 nodes and 55 hyperedges, which are figures comparable to the 2019 data, with a somewhat lower number of publications reported. Once again, we make use of several interaction modes supported by Onion, including the sorting, cardinality, and related modes. The overview of the data set is rather similar to the previous year’s data, and we can find out that the hyperedge with the largest cardinality in this case also connects 6 nodes. This hyperedge corresponds to the publication by Backåberg et al. [80]—interestingly enough, the respective publication actually includes 7 co-authors, but one of them is an external collaborator with no LNU affiliation, who is therefore not allocated a node in the extracted data set.

Finally, we load both data sets in Onion and activate the comparison mode, with the results displayed in Figure 5.8. The first note to make is that the hypergraph on the left appears to include a larger number of two-node edges in the central area (which makes sense, given the larger overall number of (hyper-)edges in that data set), however, the central views of both left and right hypergraphs are perceived

[4]URL: `http://lnu.diva-portal.org/` (accessed: August 20, 2021)

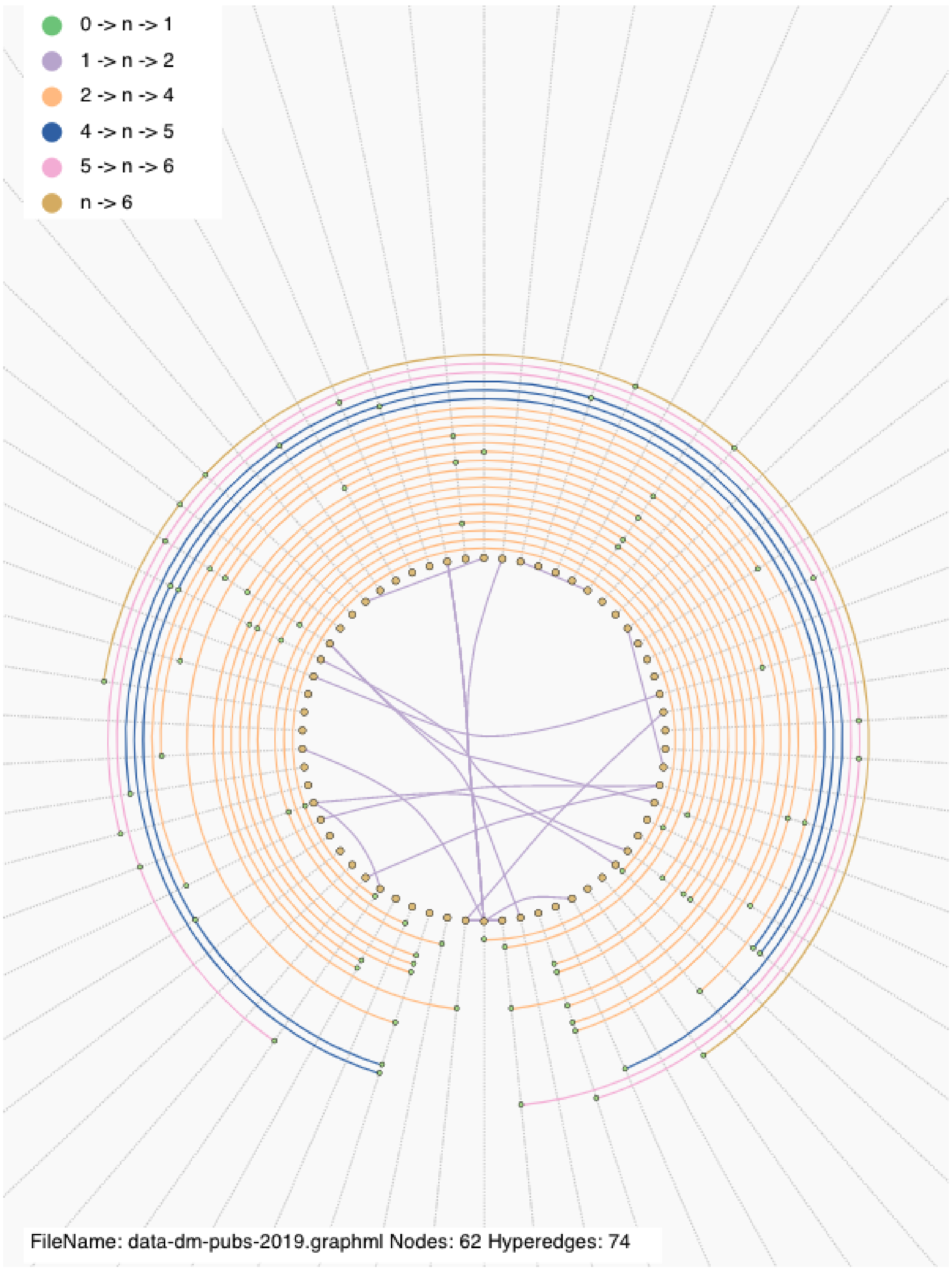


Figure 5.6: *Exploration of the 2019 publications data set. The hypergraph includes 62 nodes and 74 hyperedges.*

without severe edge cluttering issues due to the use of edge bundling. Given the nature of the data, the next question to ask is "who are the authors (i.e., nodes) missing in either data set?". The color-coding used in the comparison mode facilitates us in answering that question, and we are able to discover the authors who were either not affiliated with the department or university, or did not publish during the respective year. Despite various reasons and circumstances, a rather large part of the nodes (i.e., authors) appears in both data sets, as indicated with green-colored nodes in Figure 5.8. The final task that we could cover with the comparison mode for these

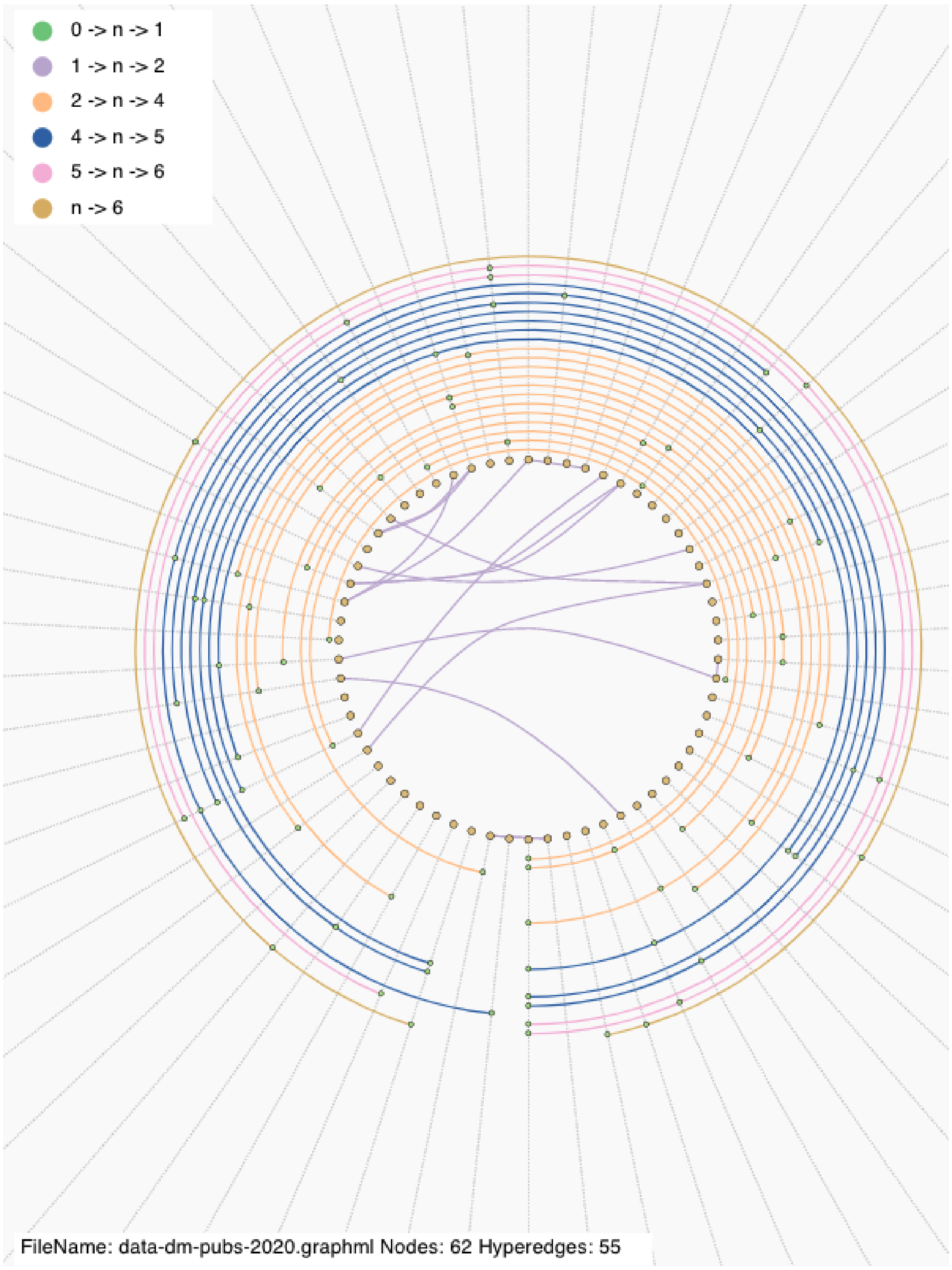


Figure 5.7: *Exploration of the 2020 publications data set. The hypergraph includes 62 nodes and 55 hyperedges.*

data sets would be "which collaborations were active in both years, and vice versa, which collaborations were only active (with regard to publications) in one of the two years?". In contrast to the previous question (related to the nodes only), only several collaboration/co-authorship configurations remain present across both years, including collaborations between (1) Kastrati and Kurti, (2) D'Angelo and Caporuscio, and (3) Hönel, Ericsson, Löwe, and Wingkvist. This is an interesting finding, which is also rather unexpected, since researchers often work and publish together in similar configurations over the course of several years.

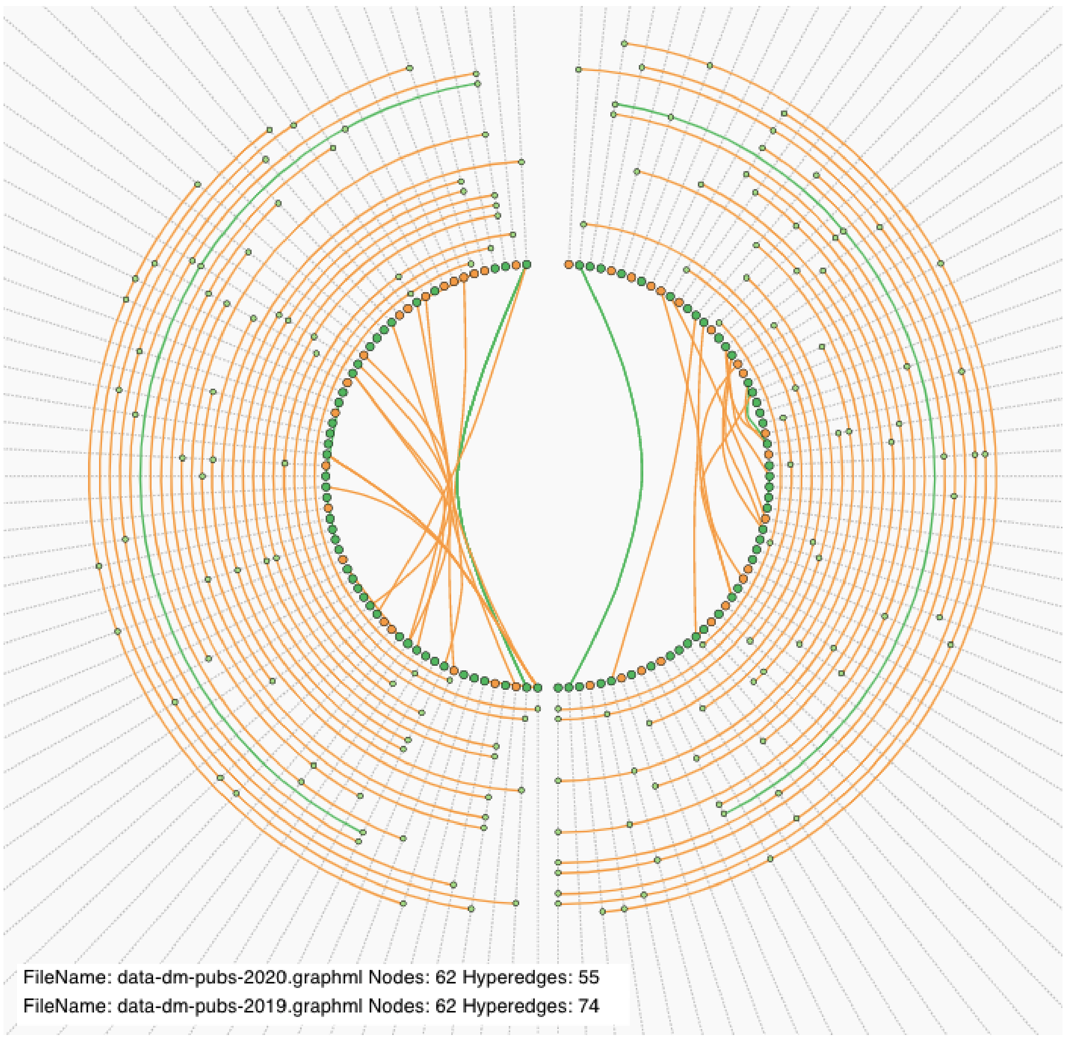

FileName: data-dm-pubs-2020.graphml Nodes: 62 Hyperedges: 55
FileName: data-dm-pubs-2019.graphml Nodes: 62 Hyperedges: 74

Figure 5.8: *Comparison of two publication data sets in Onion. The total combined number of the nodes in the data is 124, and the total number of the hyperedges is 129. It proves that the Onion tool can scale up to more than 124 nodes and 129 hyperedges.*

The final remark about this case study is related to the fact that the combined size of both data sets, as presented in the comparison mode view in Figure 5.8, is 124 nodes and 129 hyperedges. By being able to load these data sets, work with them interactively, and make observations and findings, we can confirm that the new Onion implementation is able to scale up—both with regard to the technical performance and the visual design—at least to the hypergraphs of comparable sizes and complexity.

5.2 Usability Study

In this subsection, we discuss the design and outcomes of a usability study conducted with Onion. The study was designed to collect primarily qualitative feedback from several users about the usability of the new Onion implementation, while asking them to complete several tasks and fill out several questionnaires afterwards.

5.2.1 Study Design

According to the research of Stasko's holistic set of metrics that assess the value of visualization [62], a methodology to evaluate the visualization approach is provided, literally called ICE-T [61]. ICE-T quotes the first letter from four value objectives. They are *Insight*, *Confidence*, *Essence*, and *Time*.

To investigate and assess the Onion tool with regard to usability and quality aspects, we apply the ICE-T methodology in our project. ICE-T is a heuristic-based methodology and it employs a series of low-level, detailed questions to be answered or benchmark tasks to determine the perceived value of the visualization approaches [61]. In the ICE-T questionnaire, each question receives a 7-point Likert scale score [63], and the aggregated average score of 4.0 indicates a neutral result of the visualization approach. Furthermore, a higher score is considered a positive result. Otherwise, a score lower than 4.0 means a shortcoming of the visualization. According to the requirement of ICE-T, the user study of this project involves six participants who have experience or knowledge about developing visualizations. Besides, all participants of the user study are volunteers, which means they could stop their participation and quit at any moment. Our users are three master students and three PhD students who have completed a minimum of one course on data visualization. All of the user study sessions lasted around 30 minutes. They were conducted online via Zoom, and they included 5 minutes long Onion tool introduction and 10–15 minutes operation tasks.

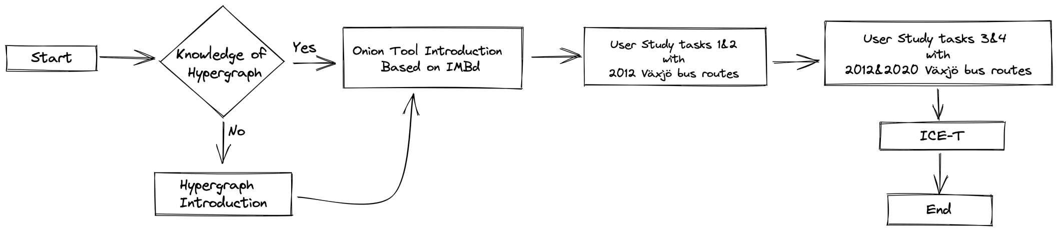

Start
Knowledge of Hypergraph
Yes
No
Onion Tool Introduction Based on IMBd
Hypergraph Introduction
User Study tasks 1&2 with 2012 Växjö bus routes
User Study tasks 3&4 with 2012&2020 Växjö bus routes
ICE-T
End

Figure 5.9: *The user study workflow.*

We have designed a user study procedure for this project as shown in Figure 5.9. Initially, we ask the users of their knowledge of hypergraphs, and explain the respective concept, if necessary. Next, we demonstrate to users some primary functions of the Onion tool via the small IMDB data set. The IMDB data set (Figure 5.10) is collected from IMDB, a well-known online database of films and actors, and recorded into GraphML files. It contains 21 actors as nodes and 16 related movies as hyperedges. Concretely, each node represents one actor and hyperedges corresponding to the movie co-actors. The user study participants are free to ask any questions about the data or the tool at this point.

After the operation illustration of the Onion tool, we get to the central part of the user study—the user operation tasks. In order to engage the users in interaction with the tool and analyze the user behavior, we determine the tasks for the same case study data, 2012 and 2020 Växjö bus routes, as in Section 5.1.1, for the four following questions:

- Task 1: Check if there is any intersection between Line 3 and Line 8, and if there is any, name it and mention if any other lines cross at the same bus stop.

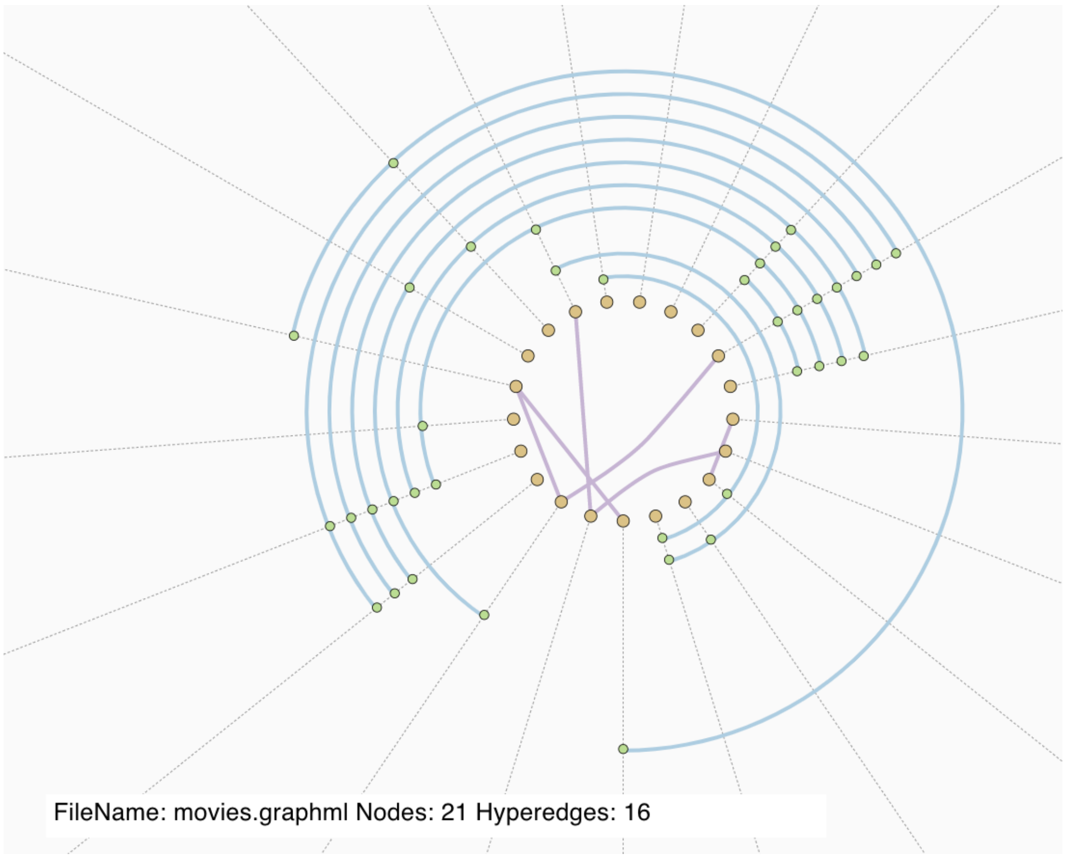


Figure 5.10: *The small movie data set fetched from IMDB. It contains 21 actors as nodes and 16 related movies as hyperedges.*

- Task 2: Find the bus stop(s) with the second largest number of bus lines intersecting, and name the bus lines missing from that stop(s).
- Task 3: Name the bus lines that are present in one of the data sets, but missing from another one.
- Task 4: Find the highest cardinality frequency in both data sets, i.e., the cardinality number that appears the most in the graphs.

Please note that while the same data set is used here as for the case study in Section 5.1.1, the user study is independent of the case study with regard to its goals, participants, procedure, and results. The motivation for choosing this particular real-world data set was that we considered it to be potentially more interesting and engaging for most participating users than academic publication/collaboration data, for instance, or synthetic hypergraph data. However, other suitable hypergraph data sets could be used for the user study instead.

For the first two tasks, the users need to focus on the single hypergraph of the 2012 Växjö bus routes. Task 1 carries out the usability of edge bundling in the context of hypergraph exploration. Task 2 focuses on finding evidence about the effectiveness and efficiency of a hypergraph comparison technique based on the Onion approach. In this task, users must quickly discover the answer through different visual modes of the Onion tool, such as related mode and sorting mode. Next, for the last two tasks, we want to test the capacity of the Onion tool under two hypergraphs. The users need to find out the answer by comparing 2012 and 2020

Växjö bus routes. We want to understand if the user can easily find the answer by comparison mode of the Onion tool in Task 3 and the performance of cardinality mode with two hypergraphs in Task 4.

The results of the usability study and the feedback provided by the participants are discussed next.

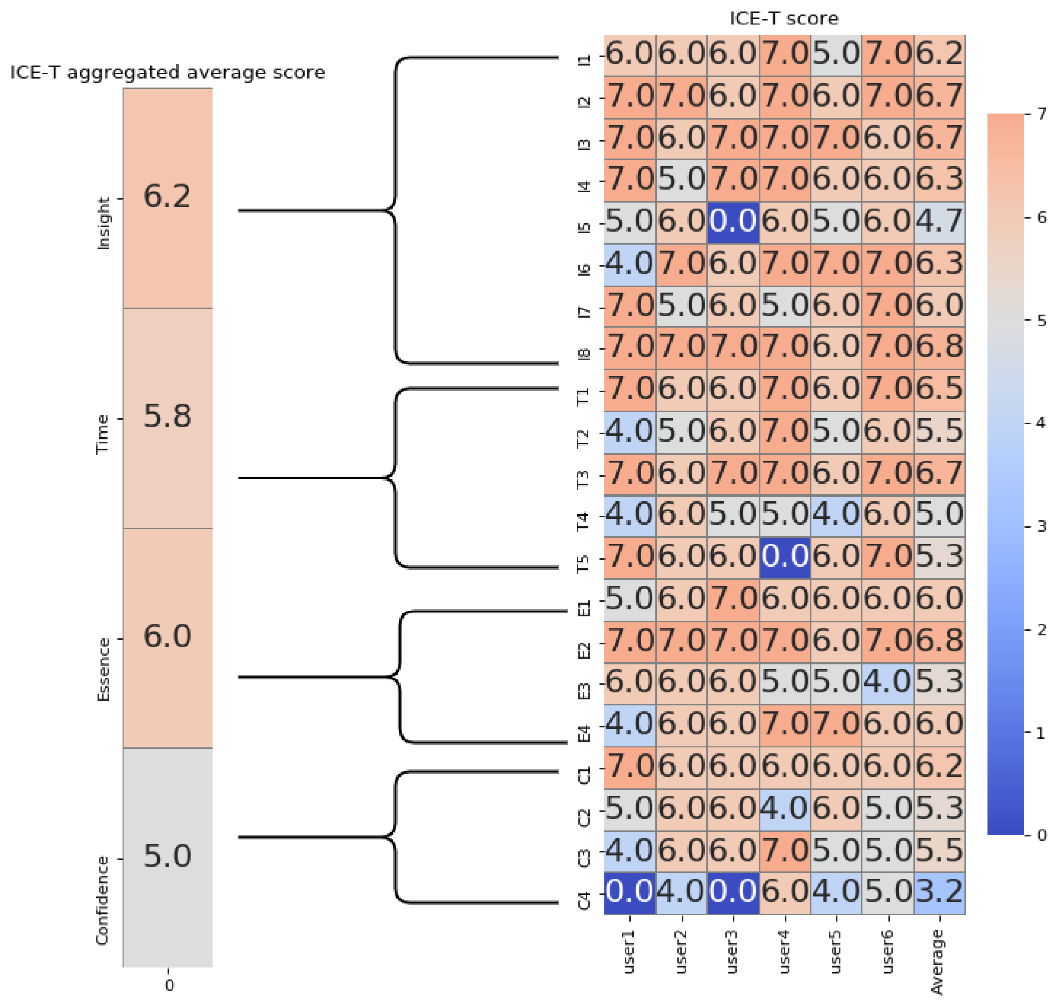
5.2.2 Study Results

During the introduction, the main problem of the Onion tool has turned out to be that the users have a hard time understanding the difference between the node and endpoint, which we had not expected. Besides, half of the users were confused with the binary/ordinary hyperedges, which occupy the inner area of the Onion tool, and the radial hyperedges. The latter issue has also contributed to another problem during the user operation phase, as we found out later.

In the user operation task, all users finished their tasks in under 12 minutes. Moreover, they have spent most of their time on Task 1. Half of the users could not find the target element during Task 1, and consequently, the responses have the lowest accuracy among the four tasks. As we mentioned, some users have a problem understanding the difference between the binary/ordinary hyperedges and the radial hyperedges. Half of the users excessively concentrated on the radial hyperedges, but ignored the existence of binary hyperedges in the inner area. The others also spend more time than we had expected in Task 1. Besides, we have discovered that the Onion tool does not effectively assist the users in finding the answer for this particular task: the related mode only highlights the hyperedges relative to the target node, but not the related nodes that are existing in the same hyperedge group. However, as a different positive outcome, we have recorded a perfect accuracy in Task 2 with a single hypergraph data file. Compared to the issues with the previous task, the related mode highlights the related nodes in the same hyperedge group and helps the users find out the missing nodes quickly, as the results of Task 2 have demonstrated. The results of Tasks 3 and 4 are also quite positive. The users have been able to answer these questions under 4 minutes using Onion. Some of the users initially misunderstood the question of Task 4, however, it did not affect the users in finding out the correct answer eventually. We would say that the performance of the visual and interactive functions of the Onion tool for this tasks was solid, as we had expected.

According to the ICE-T questionnaire results (see Figure 5.11), the cumulative average score of our Onion tool is 5.75, which means it is a decent visualization score. However, we think there is still a long way to us to achieve a higher score. Analyzing the aggregated average score of the ICE-T, each section of the ICE-T test is analyzed independently to detect opinions for each metric [61], and it shows that the average value of *Confidence* (as defined by ICE-T) is the lowest within a passing level. Digging into the average score of confidence, we determine that the lowest scores were provided for the following question: *The visualization helps understand data quality—If there were data issues like unexpected, duplicate, missing, or invalid data, the visualization would highlight those issues.*

The low score for this question can be considered in relation to the current Onion tool limitations. Since the space of the inner area is small, all users have mentioned that the binary hyperedges can not be recognized. When many binary hyperedges exist in the inner area, the edge bundling cannot offer a straightforward observation experience for the users to target the element. In contrast, it results in deteriorated performance and user experience, so that the users had to turn off the edge bundling functionality. Next, each group's color of the cardinality mode is similar, and each group of the cardinality mode does not give the users an accurate number of the cardinality. The above problems lead the users to erroneous data. The discussion of the limitations of the current implementation continues below.



	user1	user2	user3	user4	user5	user6	Average
I1	6.0	6.0	6.0	7.0	5.0	7.0	6.2
I2	7.0	7.0	6.0	7.0	6.0	7.0	6.7
I3	7.0	6.0	7.0	7.0	7.0	6.0	6.7
I4	7.0	5.0	7.0	7.0	6.0	6.0	6.3
I5	5.0	6.0	0.0	6.0	5.0	6.0	4.7
I6	4.0	7.0	6.0	7.0	7.0	7.0	6.3
I7	7.0	5.0	6.0	5.0	6.0	7.0	6.0
I8	7.0	7.0	7.0	7.0	6.0	7.0	6.8
T1	7.0	6.0	6.0	7.0	6.0	7.0	6.5
T2	4.0	5.0	6.0	7.0	5.0	6.0	5.5
T3	7.0	6.0	7.0	7.0	6.0	7.0	6.7
T4	4.0	6.0	5.0	5.0	4.0	6.0	5.0
T5	7.0	6.0	6.0	0.0	6.0	7.0	5.3
E1	5.0	6.0	7.0	6.0	6.0	6.0	6.0
E2	7.0	7.0	7.0	7.0	6.0	7.0	6.8
E3	6.0	6.0	6.0	5.0	5.0	4.0	5.3
E4	4.0	6.0	6.0	7.0	7.0	6.0	6.0
C1	7.0	6.0	6.0	6.0	6.0	6.0	6.2
C2	5.0	6.0	6.0	4.0	6.0	5.0	5.3
C3	4.0	6.0	6.0	7.0	5.0	5.0	5.5
C4	0.0	4.0	0.0	6.0	4.0	5.0	3.2

Figure 5.11: *The data that represents the average and detailed score of the ICE-T questionnaires filled out by the user study participants. Aggregated average scores are presented on the left hand side. The detailed responses are presented in the table on the right. Each table row is the users' question score, while each column points to a single participant. The final column is the aggregated average for each question. The value of 0 means the user has not provided an answer to the question. The values 1–7 correspond to a Likert scale, with 4 indicating a neutral result.*

5.3 Discussion

In this subsection, we discuss the computer performance of the Onion tool and the results of the research questions based on the data gathered.

5.3.1 Limitations and Scalability

Our own experiences and the short evaluation cannot provide concrete evidence to identify a maximum number of binary/ordinary hyperedges that the Onion tool can handle before the view becoming too cluttered, too small, and unreadable for the user. One observation that we can reinforce is that the capabilities of edge bundling are restricted when the visible drawing area of the inner circle is small, for instance, when loading a hypergraph with a small number of nodes and a large number of binary/ordinary hyperedges. According to the results we have obtained, we can conclude that the edge bundling approach brings rather negative user feedback with binary hypergraph exploration in the new version of Onion in most scenarios.

The support for comparison of several hypergraphs of the new version of Onion is effective and efficient. We have receive positive feedback from both case and user studies. The users can instantly find out the difference between two hypergraphs with the Onion tool. We want to point out that all users in the user study finished the respective tasks (Tasks 3 and 4) under 4 minutes. However, as we mentioned above, using the Onion cardinality mode, the colors of the cardinality groups are similar; thus could drive the users to erroneous conclusions in certain scenarios, and could thus be considered an opportunity for further improvements of the visual encoding.

According to the data we observed from the *Chrome* browser inspector, when the loaded hypergraphs have more than 120 nodes and 120 hyperedges, the *heap size in use by the live JavaScript object* stays under 16.8 MB. The *heap size trend in the last two minutes* can increase to 15.1 MB/s at the beginning of the loading processes. Then the *heap size trend over in the last two minutes* drops to approximately 1.2 KB/s after 30 seconds. Moreover, we cannot see an apparent time lag reaction in the Onion tool during the user operation. We found evidence that the new version of the Onion tool can scale to larger hypergraphs concerning computational performance and usability, as demonstrated in the case studies. Based on our experiences and the evaluation results, we believe that the new version of the Onion tool scales well up to 150 nodes and 200 hyperedges.

5.3.2 Comparison with Previous Approaches

Considering and comparing with existing approaches, the new version of the Onion tool has its pros and cons. We will review these considerations in this subsection.

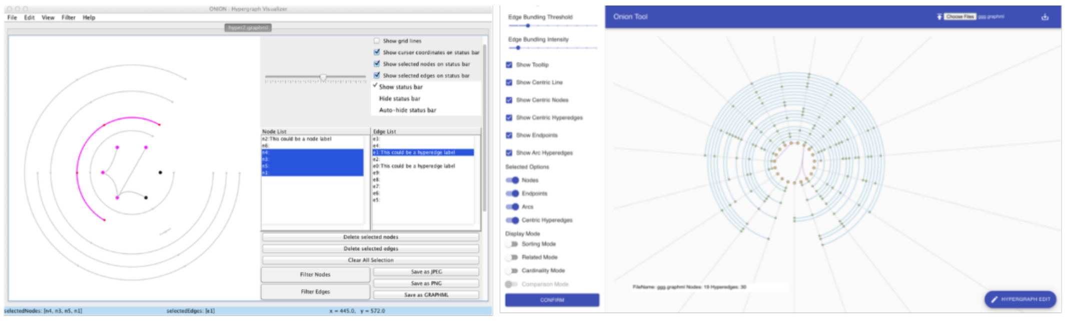


Figure 5.12: *The original (left) and the new (right) implementations of the interactive Onion approach.*

First of all, we start with a comparison with the original Onion implementation [12]. Figure 5.12 (left) presents the original tool, which was developed in Java.

The right side of the figure presents the new Onion tool that was developed from scratch using JavaScript and web-based technologies for this thesis project. In the new Onion tool, besides using the same visual metaphor and following the main design principles described for the original approach, as mentioned in Section 1.1.4, we have aimed to address the previously identified challenges and introduce further functionality to improve the usability of this visualization approach.

The major new contribution is the support for several interactive visual modes that provide visual assistance to the user in exploring the data (see Section 4.4.8), including the support for visual comparison of two hypergraphs, which was not available in the original Onion implementation and is supported by few of the existing hypergraph visualization approaches. Related to this improvement, we can also mention support for visual encoding adjustments and interactions for hiding, displaying, disabling, and highlighting of hypergraph elements (as discussed in relation to the control panel in Section 4.4.5) and viewing further details on demand via tooltips (see Section 4.4.3). Support for the control panel as well as data editing and filtering functionality with table views (see Section 4.4.6) has also been implemented while taking the visual scalability into account. For instance, the old tool would use a significant portion of the available screen space for various controls, while the proposed implementation allows these panels and dialogs to be hidden/closed in order to focus to the main view.

Furthermore, the new version of the Onion tool can easily handle more than 120 nodes and 120 hyperedges, which are values larger than reported by the resuls of the original study [12], and only considered as potentially possible at that time. In our opinion, it is more potent than many other hypergraph approaches (see Table 5.1), especially concerning the number of hyperedges. Additionally, supporting the GraphML format [23], the Onion tool offers an flexible data editing function for a single hypergraph data set. The users can change the data with simple operations. Moreover, as mentioned above, the new Onion tool supports the comparison of two hypergraphs effectively and efficiently, and none of the previous hypergraph approaches do support this functionality.

Of course, the Onion tool has some obvious shortcomings, too. It does not have support for dynamic/temporal hypergraphs, and thus, the users cannot observe the changing of the hypergraph or predict data behavior over time, in contrast to the approaches such as Hyper-Matrix [73] and PAOH [68]. However, this was not part of the requirements for our project. Other shortcomings identified for the current implementation are that the Onion tool still cannot effectively solve the edge crossing and clutter in the inner area, even with the edge bundling approach implemented. Addressing these issues could be considered part of the future work.

Table 5.1: Comparison of the hypergraph visualization approaches.

Property	**Onion (new)**	**Hyper-Matrix**	**PAOH**	**Traditional approaches**
Visual scalability	Yes	Yes	Yes	No
Data editing	Yes*	Yes*	No	No
Edge crossing issues	Yes	No	No	Yes
Hypergraphs comparison	Yes	No	No	No
Interaction	Yes	Yes	Yes	No
Timeline	No	Yes	Yes	No
Data prediction	No	Yes	No	No

6 Conclusions and Future Work

In this project, we focused on the investigation of visual representation and interactive analysis of hypergraphs. Based on the previous study of the Onion representation, we implemented an entirely new web-based Onion tool that follows the respective visual metaphor. In order to address the shortcomings and potential extensions of the previous Onion tool implementation, we made the respective changes and improvements for our proposed visualization approach, such as the scalability improvements, hypergraph data editing support, and support for visual comparison of hypergraphs. To complete this project, independent research efforts were required in relation to 1) analysis of the existing visualization, interaction, and evaluation techniques, 2) design and implementation of interactive web-based visualizations, 3) design and realization of several evaluation approaches (namely, case studies and user studies), and 4) analysis of the results, reflection, and critical discussion.

In this section, we briefly summarize the results of project work in relation to the research questions formulated in the beginning of the report. Moreover, we discuss what we are looking forward to as part of our future study of the Onion hypergraph visualization approach.

6.1 Project Summary

In this subsection, we will shortly discuss the answers to the research questions formulated in Section 1.4, based on the results gathered throughout the project.

- **RQ1** Does the edge bundling approach implemented in the new version of Onion for binary edges facilitate hypergraph exploration?

  Both case and user studies show that the usability of the edge bundling approach is rather poor. The Onion inner area restricts the performance of the binary/ordinary hyperedges. In limited space, the capabilities of hyperedge observation are restricted. Additionally, since all binary/ordinary hyperedges have the same color due to the same cardinality value, bundling makes them more challenging for users to differentiate and pick the target they want.

- **RQ2** Can the interactive visualization approach implemented in the new version of Onion support the comparison of several hypergraphs effectively and efficiently?

  Yes, we have received positive feedback from our studies regarding the comparison function of the new Onion tool. The users can instantly find out the difference between two hypergraphs using the Onion tool. Based on the case and user studies, and considering the ability to solve the comparison tasks within reasonable time using the implemented comparison approach, we can conclude that our Onion tool supports the comparison of several hypergraphs effectively and efficiently [27].

- **RQ3** How does the interactive visualization approach implement in the new version of Onion scale to larger hypergraphs with regard to computational performance and usability?

  As we have discussed in this thesis report, the new web-based Onion tool implementation was able to handle hypergraphs with more than 120 nodes and

120 hyperedges, with rather low values of the random memory usage reported by the web browser compared to the typical specifications of the modern computers. Besides memory consumption, it is important to notice that the visual interface of the tool remained responsive and no evident lags were noted either by us, or by the users participating in the evaluation. Regarding usability, we have collected evidence that the new Onion tool has received positive user feedback and demonstrated rather good usability qualities, albeit the scale of our evaluation was limited to two case studies and a single user study with six participants.

6.2 Future Work

In this project, we have presented an entirely new version of the Onion approach for interactive visual analysis of hypergraphs. However, it is still not perfect. The primary purpose of the Onion tool is to facilitate exploration of hypergraphs, while addressing the issues of edge crossing and clutter. However, we have found evidence that edge bundling is not a perfect solution to the above problem. Here, the performance of the binary hyperedges is strongly restricted by the inner area range.

For future work, we can consider moving the binary/ordinary hyperedge representations outside of the inner area. However, this solution will raise the issue that the inner area will not be effectively used. Thus, we still want to find out how to effectively use the limited display space of the Onion without any edge crossing for ordinary edges. Secondly, the color of the cardinality group is the other issue we want to solve. We plan to improve the current color of the Onion to make the element becomes recognizable. Otherwise, the user can lose his/her mental map [81]. Furthermore, the currently implemented comparison mode affects the layout of the hypergraph visualization, and thus we could consider comparing it to a pure juxtaposition approach [26], i.e., displaying two independent Onion representations side by side as part of the comparison approach. In this case, the effectiveness and efficiency of both approaches for hypergraph comparison tasks could be evaluated with a controlled user study [34]. Finally, improvements for the related mode of the Onion tool are also part of future work. Since the related mode of the Onion only highlights the hyperedges relative to the target node, but not the related nodes that are existing in the same hyperedge group, we need to improve the respective implementation in the future.

References

[1] M. E. J. Newman, *Networks: An Introduction.* Oxford University Press, 2010.

[2] S. K. Card, J. D. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann, 1999.

[3] J.-D. Fekete, J. J. van Wijk, J. T. Stasko, and C. North, "The value of information visualization," in *Information Visualization: Human-Centered Issues and Perspectives*, ser. LNCS. Springer, 2008, vol. 4950, pp. 1–18.

[4] D. A. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual analytics: Definition, process, and challenges," in *Information Visualization: Human-Centered Issues and Perspectives*, ser. LNCS. Springer, 2008, vol. 4950, pp. 154–175.

[5] D. A. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, Eds., *Mastering the Information Age: Solving Problems with Visual Analytics.* Eurographics Association, 2010.

[6] R. Tamassia, Ed., *Handbook of Graph Drawing and Visualization.* Chapman & Hall/CRC, 2016.

[7] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner, "Visual analysis of large graphs: State-of-the-art and future research challenges," *Computer Graphics Forum*, vol. 30, no. 6, pp. 1719–1749, 2011.

[8] C. D. Correa, "History and evolution of social network visualization," in *Encyclopedia of Social Network Analysis and Mining.* Springer, 2017.

[9] M. M. Wolf, A. M. Klinvex, and D. M. Dunlavy, "Advantages to modeling relational data using hypergraphs versus graphs," in *Proceedings of the IEEE High Performance Extreme Computing Conference*, ser. HPEC '16. IEEE, 2016, pp. 1–7.

[10] C. Berge, Ed., *Graphs and Hypergraphs.* North-Holland Pub. Co., 1973.

[11] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," *VLSI Design*, vol. 11, no. 3, pp. 285–300, 2000.

[12] A. Kerren and I. Jusufi, "A novel radial visualization approach for undirected hypergraphs," in *Short Papers of the EG/VGTC Conference on Visualization*, ser. EuroVis '13. The Eurographics Association, 2013.

[13] U. Brandes, S. Cornelsen, B. Pampel, and A. Sallaberry, "Path-based supports for hypergraphs," *Journal of Discrete Algorithms*, vol. 14, pp. 248–261, 2012.

[14] G. Bachman, *Introduction to p-adic Numbers and Valuation Theory.* Academic press, 1964.

[15] X. Ouvrard, "Hypergraphs: An introduction and review," *arXiv Preprints*, vol. abs/2002.05014, 2020.

[16] M. E. Newman, “Scientific collaboration networks. I. network construction and fundamental results,” *Physical Review E*, vol. 64, no. 1, p. 016131, 2001.

[17] O. N. Temkin, A. V. Zeigarnik, and D. Bonchev, *Chemical Reaction Networks: A Graph-Theoretical Approach*. CRC Press, 1996.

[18] C. Chauve, M. Patterson, and A. Rajaraman, “Hypergraph covering problems motivated by genome assembly questions,” in *International Workshop on Combinatorial Algorithms*. Springer, 2013, pp. 428–432.

[19] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, “Multilevel hypergraph partitioning: Applications in VLSI domain,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, pp. 69–79, 1999.

[20] H. Zhou, P. Xu, X. Yuan, and H. Qu, “Edge bundling in information visualization,” *Tsinghua Science and Technology*, vol. 18, no. 2, pp. 145–156, 2013.

[21] C. Ware, *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., 2004.

[22] B. Yost and C. North, “The perceptual scalability of visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 837–844, 2006.

[23] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall, “GraphML progress report structural layer proposal,” in *International Symposium on Graph Drawing*. Springer, 2001, pp. 501–512.

[24] B. Shneiderman, “The eyes have it: A task by data type taxonomy for information visualizations,” in *Proceedings of the IEEE Symposium on Visual Languages*, ser. VL ’96, 1996, pp. 336–343.

[25] S. Zhang, Z. Ding, and S. Cui, “Introducing hypergraph signal processing: Theoretical foundation and practical applications,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 639–660, 2019.

[26] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts, “Visual comparison for information visualization,” *Information Visualization*, vol. 10, no. 4, pp. 289–309, 2011.

[27] E. Frøkjær, M. Hertzum, and K. Hornbæk, “Measuring usability: Are effectiveness, efficiency, and satisfaction really correlated?” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’00. ACM, 2000, pp. 345–352.

[28] S. G. Eick and A. F. Karr, “Visual scalability,” *Journal of Computational and Graphical Statistics*, vol. 11, no. 1, pp. 22–43, 2002.

[29] C. Andrews, A. Endert, B. Yost, and C. North, “Information visualization on large, high-resolution displays: Issues, challenges, and opportunities,” *Information Visualization*, vol. 10, no. 4, pp. 341–355, 2011.

[30] A. Bretto, “Applications of hypergraph theory: A brief overview,” *Hypergraph Theory*, pp. 111–116, 2013.

[31] M. Adamski, M. Wiśniewska, R. Wiśniewski, and Ł. Stefanowicz, "Application of hypergraphs to the reduction of the memory size in the microprogrammed controllers with address converter," *Przegld Elektrotechniczny*, no. 8, pp. 134–136, 2012.

[32] L. Leinweber and S. Bhunia, "Fine-grained supply gating through hypergraph partitioning and Shannon decomposition for active power reduction," in *Proceedings of the Design, Automation and Test in Europe Conference*, ser. DATE '08. IEEE, 2008, pp. 373–378.

[33] T. Munzner, *Visualization Analysis & Design*. CRC Press/Taylor & Francis Group, 2015.

[34] H. C. Purchase, *Experimental Human-Computer Interaction: A Practical Guide with Visual Examples*. Cambridge University Press, 2012.

[35] N. Elmqvist and J. S. Yi, "Patterns for visualization evaluation," *Information Visualization*, vol. 14, no. 3, pp. 250–269, 2015.

[36] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '14. ACM, 2014, pp. 38:1–38:10.

[37] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale, "Empirical studies in information visualization: Seven scenarios," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 9, pp. 1520–1536, 2012.

[38] T. Isenberg, P. Isenberg, J. Chen, M. Sedlmair, and T. Möller, "A systematic review on the practice of evaluating visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2818–2827, 2013.

[39] J.-D. Fekete and J. Freire, "Exploring reproducibility in visualization," *IEEE Computer Graphics and Applications*, vol. 40, no. 5, pp. 108–119, 2020.

[40] Y. Huang, Q. Liu, and D. Metaxas, "Video object segmentation by hypergraph cut," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '09. IEEE, 2009, pp. 1738–1745.

[41] S. Klamt, U.-U. Haus, and F. Theis, "Hypergraphs and cellular networks," *PLoS Computational Biology*, vol. 5, no. 5, p. e1000385, 2009.

[42] D. Gibson, J. Kleinberg, and P. Raghavan, "Clustering categorical data: An approach based on dynamical systems," *The VLDB Journal*, vol. 8, no. 3, pp. 222–236, 2000.

[43] U. Feige, J. H. Kim, and E. Ofek, "Witnesses for non-satisfiability of dense random 3CNF formulas," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '06. IEEE, 2006, pp. 497–508.

[44] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, "On the desirability of acyclic database schemes," *Journal of the ACM (JACM)*, vol. 30, no. 3, pp. 479–513, 1983.

[45] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Advances in Neural Information Processing Systems*, vol. 19, pp. 1601–1608, 2006.

[46] P. E. Black, "Dictionary of algorithms and data structures," 1998, accessed: August 20, 2021. [Online]. Available: https://xlinux.nist.gov/dads/

[47] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[48] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie, "Beyond pairwise clustering," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 838–845.

[49] Q. Fang, J. Sang, C. Xu, and Y. Rui, "Topic-sensitive influencer mining in interest-based social media networks via hypergraph learning," *IEEE Transactions on Multimedia*, vol. 16, no. 3, pp. 796–812, 2014.

[50] D. Li, Z. Xu, S. Li, and X. Sun, "Link prediction in social networks based on hypergraph," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 41–42.

[51] L.-E. Martinet, M. Kramer, W. Viles, L. Perkins, E. Spencer, C. Chu, S. Cash, and E. Kolaczyk, "Robust dynamic community detection with applications to human brain functional networks," *Nature communications*, vol. 11, no. 1, pp. 1–13, 2020.

[52] E. H.-H. Chi, *A Framework for Information Visualization Spreadsheets*. University of Minnesota, 1999.

[53] W. Schroeder, K. Martin, L. Avila, and C. Law, *The VTK User's Guide*. Kitware, 1999.

[54] A. Jackson, J. Lin, I. Milligan, and N. Ruest, "Desiderata for exploratory search interfaces to web archives in support of scholarly activities," in *Proceedings of the IEEE/ACM Joint Conference on Digital Libraries*, ser. JCDL '16. IEEE, 2016, pp. 103–106.

[55] H. C. Purchase, "Which aesthetic has the greatest effect on human understanding?" in *International Symposium on Graph Drawing*. Springer, 1997, pp. 248–261.

[56] H. C. Purchase, D. Carrington, and J.-A. Allder, "Empirical evaluation of aesthetics-based graph layout," *Empirical Software Engineering*, vol. 7, no. 3, pp. 233–255, 2002.

[57] W. Huang and P. Eades, "How people read graphs," in *Proceedings of the Asia-Pacific Symposium on Information Visualisation*, ser. APVis '05, vol. 45. Australian Computer Society, Inc., 2005, pp. 51–58.

[58] W. Huang, S.-H. Hong, and P. Eades, “Predicting graph reading performance: A cognitive approach,” in *Proceedings of the Asia-Pacific Symposium on Information Visualisation*, ser. APVis ’06, vol. 60. Australian Computer Society, Inc., 2006, pp. 207–216.

[59] W. Huang, P. Eades, and S.-H. Hong, “Beyond time and error: A cognitive approach to the evaluation of graph drawings,” in *Proceedings of the Workshop on BEyond Time and Errors: Novel EvaLuation Methods for Information Visualization*, ser. BELIV ’08. ACM, 2008.

[60] W. Huang, P. Eades, and S.-H. Hong, “Measuring effectiveness of graph visualizations: A cognitive load perspective,” *Information Visualization*, vol. 8, no. 3, pp. 139–152, 2009.

[61] E. Wall, M. Agnihotri, L. Matzen, K. Divis, M. Haass, A. Endert, and J. Stasko, “A heuristic approach to value-driven evaluation of visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 491–500, 2019.

[62] J. Stasko, “Value-driven evaluation of visualizations,” in *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*, ser. BELIV ’14, 2014, pp. 46–53.

[63] A. Joshi, S. Kale, S. Chandel, and D. Pal, “Likert scale: Explored and explained,” *Current Journal of Applied Science and Technology*, vol. 7, no. 4, pp. 396–403, Feb. 2015.

[64] C. Nobre, D. Wootton, L. Harrison, and A. Lex, “Evaluating multivariate network visualization techniques using a validated design and crowdsourcing approach,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI ’20. ACM, 2020.

[65] D. Holten, “Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, 2006.

[66] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li, “Geometry-based edge clustering for graph visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1277–1284, 2008.

[67] D. Holten and J. J. Van Wijk, “Force-directed edge bundling for graph visualization,” in *Computer Graphics Forum*, vol. 28, no. 3. Wiley Online Library, 2009, pp. 983–990.

[68] P. Valdivia, P. Buono, C. Plaisant, N. Dufournaud, and J.-D. Fekete, “Analyzing dynamic hypergraphs with parallel aggregated ordered hypergraph visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 1, pp. 1–13, 2019.

[69] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers, “The state-of-the-art of set visualization,” in *Computer Graphics Forum*, vol. 35, no. 1. Wiley Online Library, 2016, pp. 234–260.

[70] K. Dinkla, M. J. Van Kreveld, B. Speckmann, and M. A. Westenberg, "Kelp diagrams: Point set membership visualization," in *Computer Graphics Forum*, vol. 31, no. 3pt1. Wiley Online Library, 2012, pp. 875–884.

[71] J. Paquette and T. Tokuyasu, "Hypergraph visualization and enrichment statistics: how the egan paradigm facilitates organic discovery from big data," in *Human Vision and Electronic Imaging XVI*, vol. 7865. International Society for Optics and Photonics, 2011, p. 78650E.

[72] N. A. Arafat and S. Bressan, "Hypergraph drawing by force-directed placement," in *Database and Expert Systems Applications*. Springer, 2017, pp. 387–394.

[73] M. T. Fischer, D. Arya, D. Streeb, D. Seebacher, D. A. Keim, and M. Worring, "Visual analytics for temporal hypergraph model exploration," *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[74] C. Berge, *Hypergraphs: Combinatorics of Finite Sets*. Elsevier, 1984, vol. 45.

[75] J. Lazar, *User-Centered Web Development*. Jones & Bartlett Learning, 2001.

[76] J. Heer, F. Van Ham, S. Carpendale, C. Weaver, and P. Isenberg, "Creation and collaboration: Engaging new audiences for information visualization," in *Information Visualization*. Springer, 2008, pp. 92–133.

[77] N. Gershon and S. G. Eick, "Information visualization," *IEEE Computer Graphics and Applications*, vol. 17, no. 04, pp. 29–31, 1997.

[78] D. Zielasko, B. Weyers, B. Hentschel, and T. W. Kuhlen, "Interactive 3D force-directed edge bundling," in *Computer Graphics Forum*, vol. 35, no. 3. Wiley Online Library, 2016, pp. 51–60.

[79] A. Alissandrakis, N. Reski, M. Laitinen, J. Tyrkkö, J. Lundberg, and M. Levin, "Visualizing rich corpus data using virtual reality," *Studies in Variation, Contacts and Change in English*, vol. 20, 2019.

[80] S. Backåberg, A. Hellström, C. Fagerström, A. Halling, A. Lincke, W. Löwe, and M. Ekstedt, "Evaluation of the skeleton avatar technique for assessment of mobility and balance among older adults," *Frontiers in Computer Science*, vol. 2, p. 55, 2020.

[81] K. Misue, P. Eades, W. Lai, and K. Sugiyama, "Layout adjustment and the mental map," *Journal of Visual Languages & Computing*, vol. 6, no. 2, pp. 183–210, 1995.