Linköping University | Department of Science and Technology
Master's thesis, 30 ECTS | Media Technology
2026 | LiU-ITN-TEK-A--26/001--SE

ViNCent 2.0: A Web-Based Implementation for the Visualization and Analysis of Multivariate and Temporal Networks

ViNCent 2.0: En webbaserad implementering för visualisering och analys av multivariata och dynamiska nätverk

Adrian Szuter

Supervisor : Andreas Kerren
Examiner : Kostiantyn Kucher

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

Network visualization plays a crucial role in understanding complex relationships across different domains, from social interactions to biological systems. However, as networks grow in size and complexity, traditional visualization approaches struggle to support the simultaneous analysis of multiple measures and temporal evolution. This thesis addresses these challenges by reimplementing and extending ViNCent, a specialized tool for centrality focused network analysis, as a modern web-based system built on a client-server architecture with a TypeScript/React/D3.js frontend and Python/FastAPI/NetworkX backend, supporting multivariate and temporal network exploration and analysis.

ViNCent 2.0 preserves the original system's core analytical capabilities while introducing significant extensions. The circular diagram view extends the original centrality focused encoding to support arbitrary node attributes alongside centrality measures, enabling simultaneous comparison of multiple centrality metrics and domain-specific metadata within a unified visual representation. For temporal networks, the system implements a discrete snapshot model with foresighted layouts, chronological node ordering and explicit change indicators to support temporal exploration while maintaining spatial consistency to reduce cognitive load during navigation.

Evaluation through a formative user study and structured use case scenario demonstrates that ViNCent 2.0 supports exploratory analysis of multivariate and temporal networks. The user study identified concrete usability improvements that informed design refinements, while the Delhi Metro network use case illustrates how the system enables analysts to track network evolution, identify structural hubs, and relate topological importance to domain specific metadata through coordinated views and interactive filtering. However, to what extent the system supports effective analysis for domain experts remains an open question left for future work.

The work contributes a functional web-based platform for network visualization that addresses the specific challenges of comparing multiple measures simultaneously and exploring temporal dynamics. By modernizing ViNCent's centrality focused approach and extending it to support broader analytical requirements, this thesis demonstrates how specialized visualization techniques can be adapted to contemporary web technologies while expanding their applicability to increasingly complex network analysis tasks.

Acknowledgments

I would like to thank everyone who participated in the interviews for their time and insights, which helped shape the final version of the system. I am especially grateful to my supervisor, Prof. Dr. Andreas Kerren, and my examiner, Dr. Kostiantyn Kucher, for their invaluable feedback and guidance throughout this work, which helped refine the direction of the research and implementation and significantly improved its quality.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter introduces the topic, motivation, objectives and scope of this thesis. It outlines the research questions, approach and limitations associated with developing an updated system for visual analysis.

1.1 Motivation

Network visualization is a technique that emerged alongside the growth of complex, connected data in the digital age. It allows humans to understand relationships not through raw numbers or tables, but through intuitive visual structures that transform abstract connections into visual forms, revealing patterns and hidden structures. A network is typically modeled as a graph consisting of nodes and edges, representing entities and their relationships, respectively. These networks can represent a wide range of systems, from social interactions and biological pathways to computer networks and scene graphs. However, as networks grow in size and complexity, so does the challenge of analyzing them.

One powerful approach used across domains like biology and sociology is the application of centrality measures. These are mathematical metrics that assign a score to each node based on its position in the overall network structure, with the aim of quantifying different notions of importance. Common centrality measures, such as degree, closeness, betweenness, and Eigenvector centrality, each capture different aspects and perspectives on how a node contributes to connectivity and flow within the network [1]. For instance, degree centrality looks at how many direct connections a node has, while betweenness centrality captures how often a node acts as a bridge along the shortest paths between others. In practice, no single centrality measure can fully reflect the role of a node in complex networks. As a result, centrality analysis often deals with multivariate data, where each node is associated with multiple centrality scores. Comparing these scores and understanding the interplay between them becomes increasingly difficult in large-scale networks. As more measures are added and as networks continue to grow in size and density, these analytical challenges increase significantly, resulting in traditional node-link diagrams alone becoming insufficient for making sense of these complex relationships.

To address these challenges, ViNCent [2] was developed as an interactive visual analytics tool for centrality-based network analysis. It allows users to reorder nodes, apply dynamic clustering, and visually explore relationships between various centrality dimensions [3]. Features such as histogram-based filtering and a radial network layout support the discovery of patterns and structures that would otherwise remain hidden. Despite its usefulness and importance, ViNCent is now outdated, with its original implementation built in Java using the now deprecated Prefuse library. This has severely limited its usability in modern environments, especially in web-based workflows. Furthermore, ViNCent lacks support for temporal networks, where the network structure evolves over time, nor does it support multivariate data visualization beyond centrality, both of which are increasingly important in real-world

applications. These limitations restrict its flexibility and make it harder to adapt the tool for broader use in research or applied contexts.

1.2 Aim

The aim of this Master's thesis is to reimplement and extend ViNCent as a modern, web-based visualization tool that supports interactive analysis of both multivariate and temporal networks while preserving and building upon its original core functionality.

1.3 Research questions

This thesis addresses the following questions:

RQ1: How can ViNCent's centrality analysis capabilities be modernized as a web-based visualization system?

RQ2: How can ViNCent be extended to support the visualization and analysis of multivariate node attributes beyond centrality?

RQ3: How can ViNCent be extended to represent temporal networks, enabling the exploration of structural and attribute-level changes across different time steps?

RQ4: How effective is the modernized ViNCent in visualizing and analyzing multivariate and temporal networks?

1.4 Delimitations

This thesis focuses on the design, implementation and evaluation of a visualization tool which is limited to the domain of network analysis and does not attempt to generalize findings to other types of data visualization. The evaluation is conducted without the involvement of external domain experts and instead relies on a formative user study and predefined use case scenarios. As such, the focus is on analytical support and usability within a research context rather than practical deployment. The system architecture is limited to a single server deployment designed to support a few users at a time. Scalability in the sense of handling larger and more complex network datasets is addressed analytically through the design of the visualization, but issues related to technical scalability, including multi-user load, authentication, distributed infrastructure, and data storage are outside the scope of this work. Additionally, the datasets used in the evaluation are limited to publicly available or previously studied network datasets, which may not represent the full range of real-world applications.

1.5 Approach

This thesis follows a design-implementation-evaluation workflow grounded in information visualization research. The new system called ViNCent 2.0 was developed iteratively, with design decisions informed by theory, prior work and formative feedback.

The evaluation adopts a qualitative approach, combining a use case scenario with a formative user study. The use case scenario is used to demonstrate how ViNCent 2.0 supports centrality-based, multivariate and temporal network analysis tasks. This scenario showcases a typical analytical workflow and illustrates how the proposed visual encodings and interaction features can be used in practice.

In addition, a small user study was conducted on a near final version of the system. The purpose of this study was not to provide a final validation of the tool, but rather to gain formative insights into usability, interaction design and analytical support. The findings from

this study were used to identify strengths and weaknesses of the initial design and directly informed subsequent revisions and improvements to the system.

No separate summative user evaluation was conducted on the final version of the tool. While such an evaluation was considered, it was not feasible within the scope of this thesis. Instead, the final system reflects an iterative refinement process based on user feedback and design reflection, which aligns with established design study and iterative evaluation practices in visualization research [4, 5]. The evaluation therefore focuses on demonstrating analytical capabilities and design effectiveness rather than measuring performance against predefined benchmarks.

Chapter 2
Background and Related Work

This chapter establishes the theoretical and practical foundation for the reimplementation and extension of ViNCent. It begins by introducing fundamental concepts in network analysis that motivate the need for specialized visualization tools. Building on these foundations, it reviews common network analysis tasks that inform system requirements. The original ViNCent system is then examined in detail, highlighting both its contributions to centrality focused analysis and its limitations that motivate this work. Subsequently, the chapter explores theoretical concepts underlying multivariate and temporal networks, which form the basis for the extensions. Finally, related web-based visualization platforms are surveyed, and evaluation methodologies appropriate for exploratory visualization systems are discussed.

2.1 Network Fundamentals

Networks provide a powerful abstraction for representing relationships between entities across diverse domains. From social interactions and biological pathways to transportation systems and communication networks, the graph structure, which consists of nodes and edges, enables analysis of connectivity, influence and structural patterns.

2.1.1 Graph Representation

A network is formally modeled as a graph $G = (V, E)$ where V represents the set of nodes and E represents the set of edges connecting them [1]. Edges may be directional, indicating asymmetric relationships or undirected, representing bidirectional connections. Networks may also include weights on edges, representing the strength or capacity of relationships, and attributes on both nodes and edges, encoding additional metadata such as labels, categories or numerical properties.

This mathematical abstraction enables computational analysis but introduces significant challenges for human interpretation. As networks grow in size and complexity, traditional tabular or textual representations become inadequate for revealing structural patterns, identifying anomalies or understanding the roles of individual nodes within the larger system.

2.1.2 Centrality Measures

One of the most fundamental questions in network analysis is: which nodes are most important? Answering this question requires defining what importance means in a given context, and different centrality measures capture different notions of structural significance [6]. Some of the most common are as follows:

- *Degree centrality* measures the number of direct connections a node has. Nodes with high degree are locally well-connected and often serve as hubs. In a social network, high degree individuals have many direct contacts; in a transportation network, high degree stations connect to many routes.

- *Betweenness centrality* quantifies how often a node lies on the shortest paths between other pairs of nodes. High betweenness indicates that a node acts as a bridge or bottleneck in the network's information or resource flow. Removing such nodes can significantly fragment the network.
- *Closeness centrality* measures how close a node is to all other nodes in the network based on the sum of the shortest path distances. Nodes with high closeness can reach others quickly, making them efficient broadcasters or coordinators.
- *Eigenvector centrality* assigns importance based on the quality of a node's connections rather than just their quantity. A node connected to other important nodes receives a higher score, capturing the notion of influence through association.

Importantly, these measures often disagree. A node may rank highly in degree but poorly in betweenness or vice versa. This divergence reflects the complex nature of network structures and highlights the necessity of evaluating multiple centrality dimensions rather than relying on any single metric [1].

2.1.3 Community Detection

Beyond individual node importance, many real-world networks exhibit community structure, where some groups of nodes are more densely connected internally than to the rest of the network [7]. Community structure appears across domains such as friend groups in social networks, functional modules in biological systems, thematic clusters in citation networks, and geographic regions in transportation systems.

Identifying communities serves several purposes, as it simplifies large networks by revealing organizational structure and enabling comparative analysis across networks or time periods [7]. A node's community membership also provides important context for interpreting its centrality. A node with moderate degree might still be highly important if it bridges multiple communities.

Numerous algorithms exist to detect communities, each with different definitions of optimal partitioning [8]. Modularity-based methods, such as the Louvain algorithm [9], optimize for the fraction of edges falling within communities versus those expected by chance. Label propagation [10] offers a fast alternative where nodes iteratively adopt the majority label of their neighbors. Spectral clustering uses Eigenvector analysis of the graph structure [11], while hierarchical methods reveal nested community structure at multiple scales [12].

The choice of algorithm depends on network properties, analytical goals and computational constraints. It can also be beneficial in network analysis to apply and compare multiple decomposition methods, as different approaches may highlight distinct structural features of the same network.

While these fundamental concepts provide the mathematical and algorithmic foundation for network analysis, understanding what analysts actually do with networks, the tasks they perform and questions they ask, is equally important for designing effective visualization tools. The following section examines the analytical tasks that motivate system requirements.

2.2 Network Analysis Tasks

Understanding what analysts seek to accomplish when exploring network data is important for designing effective visualization tools. Network analysis tasks vary widely depending on the domain and analytical goals, but several fundamental task categories recur across applications. This section reviews common network analysis tasks that inform the design requirements of ViNCent 2.0.

2.2.1 Task Taxonomies

Munzner's [5] task abstraction framework provides a foundational vocabulary for understanding what analysts do when exploring network data. Her model distinguishes high level actions, such as *identify*, *compare*, *summarize*, and *locate*, from targets those actions operate on, separating the why of a task from the what. Targets themselves span multiple levels of specificity: at the data level they include features such as trends, distributions and outliers while at the network level they encompass individual nodes, edges, and clusters. This gives rise to meaningful action-target pairings, such as *discover distribution*, *compare trends*, or *identify outliers*, that capture analytical intent more precisely than either component alone.

Building on this, Lee et al. [13] offer a more domain specific taxonomy that distinguishes between topology-based tasks, which focus on structural properties of the network, and attribute-based tasks, which examine the data associated with nodes and edges. Topology-based tasks include identifying connectivity patterns, finding paths, detecting clusters and analyzing overall network structure. Attribute-based tasks involve comparing attribute values, finding correlations between attributes and structure and identifying outliers based on attribute distributions. Together, these frameworks suggest that effective network visualization must support a range of actions operating across both structural and semantic dimensions of the data.

2.2.2 Centrality Focused Tasks

For centrality analysis specifically, several recurring analytical goals emerge from the literature. Analysts frequently need to identify the most important nodes according to different notions of importance [14]. This involves not only ranking nodes by individual centrality measures but also comparing rankings across multiple measures to understand where they agree or diverge. As Freeman [6] notes, different centrality measures capture fundamentally different aspects of structural importance, making multi-measure comparison essential, rather than optional.

Another common task involves detecting nodes with conflicting centrality profiles, those that rank highly on some measures but poorly on others [15]. Such nodes often reveal interesting features. For example, a node with high degree but low betweenness might be well connected locally but not essential to global information flow. Identifying these cases requires simultaneous examination of multiple centrality dimensions, which traditional node-link diagrams struggle to support effectively (see Section 2.4.1.1).

2.2.3 Multivariate & Temporal Analysis Tasks

Pretorius et al. [16] identify a comprehensive set of tasks for multivariate network analysis, building on the framework proposed by Lee et al. [13]. Their taxonomy recognizes that multivariate networks involve both structural properties and multiple attributes associated with nodes and edges, requiring tasks that can examine these dimensions individually and in combination.

The framework distinguishes four main categories of tasks. Structure-based tasks focus on topological relationships and include adjacency tasks for finding directly connected entities, accessibility tasks for identifying entities reachable through paths of varying lengths, common connection tasks for finding entities that share connections with multiple other entities, and connectivity tasks for analyzing clusters, bridges and connected components. Attribute-based tasks examine the data values associated with network elements, involving queries to find nodes or edges with specific attribute values, compute derived properties over sets of entities and identify extreme values within attribute distributions. Browsing tasks support exploratory navigation through the network by following specific paths and revisiting previously examined entities.

A key insight from Pretorius et al. is that these network specific tasks can be decomposed into lower level analytical operations: identify, determine, relocate and compare. For example, finding the node with the maximum number of adjacent entities requires identifying an entity with specific properties, identifying adjacent entities, determining a derived property, and comparing this property across entities. This decomposition reveals how complex network analysis goals emerge from combinations of simpler analytical primitives.

The framework also identifies estimation tasks, which differ fundamentally from the precise analytical tasks described above. Estimation tasks involve forming approximate characterizations of network properties through visual inspection rather than algorithmic computation. These tasks acknowledge that analysts often seek qualitative understanding rather than exact answers, especially when exploring large or complex networks.

Understanding estimation tasks aims to gain more complete knowledge of the network and include identifying clusters of highly connected nodes, characterizing groups of nodes that share common attribute values, characterizing groups based on link attributes, and determining the overall domain of attributes and their values across the network. Analysts form approximate judgments by scanning visual representations, and these judgments may include nodes that should be excluded or omit nodes that should be included. Comparison estimation tasks focus on understanding changes in networks over time, involving the identification of trends across different network snapshots and the formulation of causal explanations for observed differences. These tasks often require external domain knowledge beyond what is represented in the network itself.

Recognizing these analytical needs, particularly the challenge of comparing multiple measures simultaneously and supporting exploratory estimation tasks, the original ViNCent system was developed specifically to address centrality focused exploration through coordinated visual representations. The following section examines that system in detail.

2.3 The Original ViNCent System

To establish the foundation for the modernized system, this section provides an overview of the original ViNCent tool initially developed by Köstinger [2] and later extended by Köstinger, Kerren, Zimmer and Jusufi [3]. Understanding ViNCent's design philosophy, core features and limitations is essential for understanding what should be preserved in the reimplementation and what needs to be extended. Figure 2.1 shows an overview of the original system.

2.3.1 Design Philosophy and Core Concept

ViNCent was developed to address a specific gap in network visualization: the need to compare and analyze multiple centrality measures simultaneously within a single visual framework [2]. While many network analysis tools compute centrality metrics, few provide dedicated visual support for understanding how these measures relate to one another across the full node set. The central insight behind ViNCent is that different centrality measures reveal different aspects of node importance and these perspectives should be seen side by side rather than examined in isolation. To enable this comparison, ViNCent introduced a centrality view with a circular layout in which nodes are arranged around a circle and each node is represented not as a simple point but as a composite visual element encoding multiple centrality values simultaneously through stacked bars. This layout is what distinguishes ViNCent from general purpose network visualization tools that make use of the traditional node-link topology.

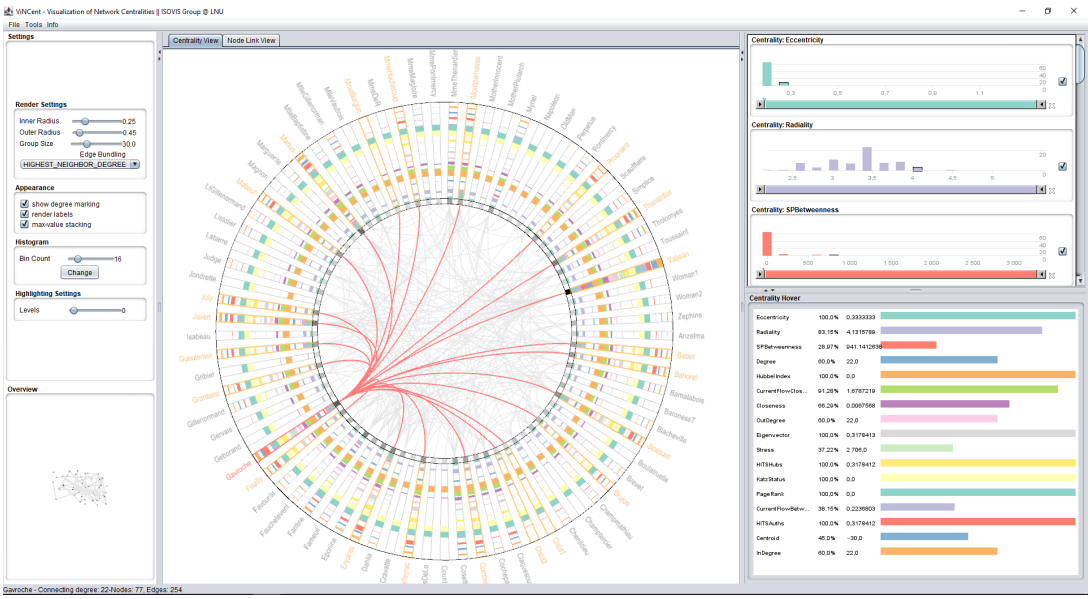


Figure 2.1: Overview of the original version of ViNCent taken by the author as a screenshot.

2.3.2 Centrality Diagram View

The centrality diagram view is the defining feature of ViNCent. In this view, nodes are positioned evenly along the circumference of a circle and each node is visualized as a radial bar extending outward from the circle's edge. Each bar is subdivided into segments, with each segment representing a different centrality measure. ViNCent supports two primary encoding modes for the bar segments. In one mode, the length of each bar segment is proportional to the value of its corresponding centrality measure, enabling direct visual comparison of magnitudes across nodes. In the alternative mode, bar segments are of uniform length, and variations in centrality magnitude are represented by the filled fraction of each segment rather than by segment length. This mode is particularly useful when the range of values varies dramatically across measures, preventing one measure from dominating the visual space.

At the center of the circular layout, network edges are drawn as curved lines connecting related nodes. This design preserves topological relationships and exploits otherwise unused interior space, while ensuring that the radial bars remain unobstructed and visually prominent. To reduce visual clutter in dense networks, the original ViNCent implemented several edge bundling modes that group related edges together. These modes use node degree and neighborhood information to determine control points for Bézier curves. The highest neighbor degree mode considers the maximum degree within each node's neighborhood, drawing edges between low-connected nodes toward the perimeter while bundling edges involving high-degree nodes toward the center. The both degrees mode uses the degrees of both connected nodes to scale control points, while the higher degree mode applies only the higher of the two degrees. Finally, the no rating mode applies uniform bundling without considering connectivity, placing all control points at fixed distances from the circle's edge.

2.3.3 Node-Link Diagram View

To complement the centrality diagram view, ViNCent includes a traditional node-link diagram that provides an intuitive representation of the network's topological structure. In this view, nodes are positioned using a force-directed layout algorithm that seeks to balance attractive and repulsive forces between nodes. Each node appears as a text label with a background, with edges drawn as straight lines between connected nodes.

2.3.4 Interactivity Analysis Features

ViNCent's interactivity is designed to support exploratory analysis. Key interactive features include the following:

- *Hover and selection*: Hovering over a node displays detailed centrality values in a tooltip and highlights the node's immediate neighbors in both views. Selection persists the highlight, allowing users to build up a focus set for comparison.
- *Zoom and pan*: Both views support standard navigation controls, enabling detailed inspection of local structures or individual nodes while maintaining the ability to return to the global overview.
- *Histogram-based filtering*: Each centrality measure is represented by a histogram showing the distribution of values across all nodes. Users can apply range filters directly on these histograms through brushing, dynamically highlighting or isolating nodes with specific centrality profiles. This allows users to dynamically focus on subsets of the network that show interesting or extreme values, making the exploration process more targeted and efficient.
- *Clustering*: ViNCent supports grouping nodes into clusters based on their centrality profiles, treating each node's set of centrality measures as a feature vector. Users can apply hierarchical or k-means clustering and visualize the resulting groups in the centrality diagram, with cluster boundaries marked visually.
- *Dynamic reordering*: Nodes in the centrality diagram can be reordered based on any centrality measure, bringing nodes with similar values into proximity. This reordering can reveal patterns such as whether high degree nodes also tend to have high Eigenvector centrality or whether betweenness and closeness rankings diverge.

Together, these features support Shneiderman's visual information seeking mantra: "overview first, zoom and filter, then details-on-demand" [17]. Users begin with a global view of all nodes and their centrality distributions, apply filters and reordering to focus on interesting subsets, and then examine individual nodes and values in detail.

2.3.5 Coordinated & Multiple Views

Coordinated & multiple views [18] form a core design principle of ViNCent, enabling users to analyze network structure and node centrality from complementary perspectives. ViNCent employs coordinated views in which the centrality and node-link diagrams are linked through interaction: hovering over or selecting a node in one view highlights the corresponding node and its connections in the other. This coordination allows users to leverage the strengths of both representations, the centrality focus of the centrality diagram and the topological layout of the node-link diagram, within a unified analytical workflow. Although only one view is fully visible at a time, a compact overview of the secondary view is always displayed to help users retain context when switching between representations.

Coordination also extends to the histogram-based filters. Histogram views are synchronized with the currently visible and filtered set of nodes, updating to reflect changes in hover state, filtering and view state. When a node is hovered over in either visualization, its corresponding position is highlighted within the histograms, reinforcing the relationship between individual nodes, their centrality values and the overall distribution.

2.3.6 Configuration and Customization

To accommodate different analytical needs and network characteristics, ViNCent provides a few configuration options accessible through a settings panel. Users can adjust both the

inner and outer radii of the circular layout to balance space allocation between centrality bars and screen space. The number of histogram bins is configurable from 2-100, allowing users to adjust the granularity of distribution visualizations based on data characteristics and analytical goals. Visual elements including degree markings and node labels can be toggled to optimize either visual clarity or performance. These settings allow users to adapt the visualization to different network sizes and display resolutions. The tool also provides options to control edge bundling behavior and select between different bundling algorithms, as described in the original work.

2.3.7 Data Import and Export

ViNCent was designed to support two data format options. The system accepts standard GraphML files [19], an open XML-based format for representing graph structures. When loading a standard GraphML file, ViNCent prompts users to select which node attributes to use for labels and which centrality measures to compute. The tool then calculates the selected measures and incorporates them into the network data accordingly.

To avoid redundant computations when reopening previously analyzed networks, ViNCent introduced a custom ViNCent GraphML format. This extended format stores computed centrality values directly as node attributes within the GraphML structure. When loading ViNCent GraphML files, users select which precomputed measures to visualize rather than triggering new calculations, significantly reducing load times for large networks with expensive centrality computations. The export functionality allows users to save their analyzed networks in ViNCent GraphML format, preserving all computed metrics for future sessions or sharing with collaborators. Additionally, visualizations can be exported as static images in various formats.

2.4 Multivariate and Temporal Networks

Many network analysis problems involve richer data than what static, single attribute models can represent. In particular, two important dimensions, temporal dynamics and multivariate attributes, create challenges for both modeling and visualization. This section introduces the theoretical foundation of temporal and multivariate networks and surveys existing network visualization tools with a focus on how they handle temporal and multivariate complexity. The goal is to highlight common strategies and identify limitations to guide the proposed system extensions.

2.4.1 Multivariate Networks

Following Kerren et al. [20], a multivariate network consists of a graph $G = (V, E)$ augmented with a set of attributes $A = \{A_1, A_2, ..., A_n\}$. For nodes, this is often expressed as a matrix $A = [a_{j,i}]$ where each row $j = 1, 2, ..., |V|$ represents a node and each column $i = 1, 2, ..., n$ an attribute. The attribute vector for a given node u is thus $a_u = (a_{u1}, a_{u2}, ..., a_{un})$ where a similar representation applies to edge attributes. These attributes can be:

- Categorical: Node type, community membership, geographic region
- Numerical: Age, weight, capacity, transaction volume
- Ordinal: Ranking, severity level, priority

The challenge is not merely to display these attributes, but to enable users to explore relationships between attributes and network structure, to identify patterns such as similar nodes connecting preferentially, to detect outliers and to understand how attributes correlate with structural roles.

2.4.1.1 Visualization Approaches for Multivariate Networks

When addressing multivariate network visualization, existing strategies tend to extend classical visualizations with techniques to encode additional attribute data. Node-link diagrams often incorporate visual variables such as color, size or shape to convey node and edge attributes [20]. However, this is typically limited to encoding only a small number of attributes before running into dimensionality issues [20]. Glyphs have also been used to represent multiple attributes, as demonstrated by Pearlman and Rheingans [21], who used compound glyphs to support security analysts in exploring multivariate data from a service-oriented perspective. However, these are useful in sparse networks but can lead to occlusion and interpretation issues in dense ones.

Matrix-based approaches represent networks as adjacency matrices, where rows and columns denote nodes and cell values indicate the presence or strength of connections. This layout avoids visual clutter and scales better than node-link diagrams in dense graphs [22]. To support multivariate analysis, matrices can be extended with heatmaps or juxtaposed attribute tables that encode node and edge attributes. For instance, systems like NodeTrix [23] integrate matrix and node-link views, while TaMax [24] overlays edge attributes directly within cells and node attributes in a juxtaposed table. These approaches reveal structural patterns and attribute correlations but often obscure topological paths, making them less effective for flow-based tasks [22]. Small multiples extend this further by presenting separate matrix or node-link views for each attribute or time period side by side, improving comparability at the cost of increased screen space [25].

More flexible and powerful are coordinated & multiple view systems, which present attributes across linked visualizations such as node-link diagrams or scatter plots, enabling complex querying and interactive filtering [18]. Systems like GraphDice presented by Bezerianos et al. [26], exemplify this approach. However, they come with drawbacks in terms of screen space consumption and increased cognitive load when coordinating between views. Additionally, while GraphDice excels at exploring attribute correlations through scatterplots, it offers limited support for analyzing the topological structure as it does not support different centrality measures nor different clustering approaches, limiting its usefulness for users more focused on the topological aspects of the network.

Ultimately, the choice of multivariate network visualization technique depends heavily on the analytical task, data complexity and user expertise. Balancing visual clarity, scalability and interactivity remain a central challenge across all approaches.

2.4.2 Temporal Networks

Temporal networks account for the evolution of relationships over time, where edges and nodes may appear, disappear or change attributes. As Holme and Saramäki [27] emphasize, understanding not just who interacts with whom, but also when those interactions occur is important for analyzing processes such as information propagation and influence. Unlike static networks, temporal networks model the sequence and timing of events. There are multiple ways to represent temporal networks depending on the level of detail required. Holme [28] distinguishes between lossless representations, which retain all information about the timing and structure of interactions, and lossy representations, which aggregate or simplify the data, often discarding temporal order. Common lossless representations include contact sequences, which are timestamped lists of interactions and, graph snapshots at discrete time intervals $G_1, G_2, ...G_T$ where each G_t represents the state of the network at time t. As Holme notes, these are theoretically equivalent in the sense that they preserve the full temporal information.

In contrast, lossy representations, such as static aggregated graphs or weighted networks, offer simpler analyses but at the cost of losing information about the order and timing of events. For example, in an email communication network, each message forms a temporal

edge between sender and receiver, and the timestamp is essential for reconstructing communication patterns. Representing such a network as a contact sequence retains complete timing of events, whereas aggregating all emails into a static graph discards the dynamics inherent in the data.

The choice between lossless and lossy representations directly impacts what can be visualized and analyzed. In a contact sequence representation, each timestamped edge can be shown on a timeline, enabling analysts to observe the exact order of interactions. The email communication network example illustrates this: if employee A emails employee B at 9:00 AM and B emails employee C at 9:15 AM, the sequence matters for understanding information flow. Aggregating these into a static graph would retain the A to B to C relationship but lose the temporal ordering that reveals B as a potential information broker in this specific cascade.

2.4.2.1 Visualization Approaches for Temporal Networks

A wide range of strategies have been proposed to support temporal network visualization and analysis, including space-time cubes [29], circular representations [30] [31] and hybrid methods that combine structural and temporal layouts [32] [33]. Among these, most can broadly be categorized into two commonly used approaches: animation-based and timeline-based [34].

Animation-based techniques, such as the one presented by Moody and McFarland [35], typically visualize change by animating node-link diagrams frame by frame, with each frame representing a different timestep. This approach is intuitive for observing gradual changes but suffers from occlusion and layout instability in dense graphs. Additionally, frequent or large layout changes across frames can disrupt users' mental maps, increasing cognitive load [36]. Techniques such as foresighted layouts [37] and edge bundling [38] mitigate some of these issues, but preserving a coherent mental map remains a challenge [36].

In contrast, timeline-based techniques provide a more stable view of temporal progression by explicitly mapping time along one axis, typically the horizontal, and organizing nodes, groups or network structures along the other. This spatial encoding supports the analysis of change across the entire time span and helps preserve users mental maps, facilitating the detection of long-term trends and temporal patterns. Examples of such approaches include Alluvial diagrams [39], which visualize group transitions over time using flowing ribbons, and Massive Sequence Views [40], which emphasize temporal activity patterns across large number of nodes. While effective for highlighting high-level structural evolution, these techniques often abstract away low-level, node to node interactions in favor of summarizing flows or aggregate patterns.

To overcome the limitations of both approaches, many modern systems adopt hybrid techniques. Systems like DyNetVis [32] and LargeNetVis [33] make use of multiple & coordinated views, where users can switch between or simultaneously view temporal and structural representations. These systems typically allow for user interaction such as brushing, filtering and zooming for exploring the networks different dimensions. However, challenges remain around maintaining visual scalability, supporting real-time exploration and preserving users mental models during transitions or filtering operations.

2.5 Existing Web-based Platforms

Beyond academic prototypes and systems, several established tools support network analysis and visualization in practice. This section reviews web-based platforms that share similar goals with ViNCent 2.0: enabling interactive exploration of network data without requiring local installation. Each tool is examined in detail to understand its strengths, limitations and how it addresses the challenges of network analysis and visualization.

2.5.1 Gephi Lite

Gephi Lite is a web-based adaptation of Gephi [41], one of the most widely used open source network analysis and visualization platforms in the research community. It provides a complete pipeline for network exploration, from data import through layout computation to visual styling and export. Users can load network data in common formats such as GEXF, GraphML and CSV edge lists directly into the browser. The tool automatically computes basic network statistics including node degree, connected components and graph density, providing immediate quantitative feedback about the networks structure.

The layout engine supports several force-directed algorithms, with ForceAtlas2 being the most prominent. This algorithm positions nodes in 2D space such that connected nodes attract each other while all nodes repel, creating layouts where dense clusters emerge visually. The implementation uses WebGL for GPU acceleration, enabling interactive frame rates for networks with thousands of nodes. Users can adjust layout parameters in real-time such as gravity strength, edge weight influence and repulsion force, allowing users to tune the visualization to emphasize different structural aspects.

Beyond layout, Gephi Lite computes a small selection of centrality measures including degree, betweenness, PageRank and HITS, as well as modular community detection. These are stored as node attributes alongside any user provided data, and the same visual mapping system applies to all of them uniformly. Numeric attributes can be mapped to node or edge size and color, while categorical attributes support color mapping only. This means a small number of attributes can be visualized simultaneously across these channels, but the approach is fundamentally limited by the number of available visual channels and there is no dedicated interface or specialized encoding for multi-measure comparison.

Gephi Lite also supports multivariate attribute exploration through a filtering panel where users can add and remove filters on nodes or edges. For numeric attributes, the panel displays a histogram of the value distribution, allowing users to define range-based filters visually. For categorical attributes, no distribution is shown, but users can select which categories to include or exclude via a dropdown. Custom JavaScript filtering scripts are also supported for more complex filtering needs. Additionally, users can add new fields or modify existing ones directly within the tool, allowing lightweight data editing without leaving the environment.

Despite these strengths, Gephi Lite does not support temporal network analysis. Networks are treated as static snapshots, and comparing network states across time requires loading separate files and mentally integrating the differences. This, combined with its limited visual channel approach to multi-measure comparison, represents a fundamental difference from ViNCent 2.0's approach of supporting simultaneous multi-measure comparison and temporal navigation within a single interface.

2.5.2 Cytoscape Web

Cytoscape Web powers web-based network analysis through the JavaScript library Cytoscape.js, which emphasizes biological networks and hierarchical data [42]. The platform originated in systems biology and has evolved into a large ecosystem supporting both molecular interaction networks and general graph visualization.

Cytoscape Web implements a diverse set of layout algorithms beyond simple force-directed approaches, including hierarchical layouts for directed acyclic graphs, circular layouts, grid layouts and specialized biological pathway layouts. Users can select layouts from a dropdown menu and apply them to the entire network or selected subgraphs, with adjustable parameters through side panels.

The platform includes a more expansive visual mapping system than Gephi Lite, allowing attributes to be mapped not only to node and edge size and color, but also to border styles, node forms such as stars or rectangles, line types such as dashed or dotted edges, and other

rendering properties. This is achieved through three mapping types: passthrough mappings that directly connect values to properties such as node labels, discrete mappings for categorical attributes, such as mapping protein type to color, and continuous mappings that associate numerical values with visual properties along a gradient. Users can define complex style rules where multiple attributes are encoded simultaneously across different visual channels, though the approach remains limited by the number of available channels and still requires manual coordination when comparing multiple measures.

Network analysis capabilities in terms of centrality measures or community detection are not included in the platform by default. These measures must be computed through separate external scripts and then imported as node or edge attributes. The platform provides a search functionality for locating specific nodes or edges by ID, but does not support filtering by attribute or values or distributions, making multivariate exploration more constrained than in Gephi Lite despite the broader visual mapping options.

Cytoscape Web does not support temporal network analysis. Dynamic networks can be handled by loading separate network files for each time point sequentially. However, maintaining visual continuity across timesteps is left to the user to manage manually.

Cytoscape Web excels at biological network analysis with strengths in layout diversity and visual customization. However, it is not optimized for centrality focused exploratory analysis. For analysts whose primary goal is comparing how different centrality measures rank, Cytoscapes interface requires a fragmented workflow: compute measures separately, load them into the platform, map them to visual channels, and mentally integrate the results. The platform offers no dedicated support for simultaneous multi-measure comparison or temporal navigation.

2.5.3 Polinode

Polinode is a commercial platform focused specifically on organizational network analysis and social network analysis in business and institutional contexts [43]. It targets HR professionals, organizational consultants and managers seeking to understand informal networks, collaboration patterns and influence structures within organizations.

Polinode differentiates itself from the others through integrated survey-based data collection. Users can create custom surveys asking employees questions like "Who do you go to for technical advice?" and Polinode automatically converts responses into network data. The platform also supports importing data from CSV files, HR systems and communication platforms like Slack and Microsoft Teams, allowing it to construct networks from actual communication patterns.

Once a network is loaded, users can choose to import and compute a wide range of network measures. These include basic descriptive statistics such as network size or density, centrality measures, community detection and clustering metrics. Each measure is accompanied by plain language explanations which helps non-expert users interpret what each measure implies.

In terms of visual mapping, Polinode sits between Gephi Lite and Cytoscape Web in terms of flexibility. Colors and sizes can be mapped to nodes and edges, and node shapes provide an additional visual channel for distinguishing categories, offering slightly more customization than Gephi Lite. However, it does not support the broader rendering options available in Cytoscape Web such as border styles or line types. Filtering operations are supported allowing users to highlight specific subsets of the network by attribute, but no distributions are shown and custom filtering scripts are not supported.

Polinode also provides tabular views in the form of collaboration and membership matrices, which can be displayed side by side with the node-link diagram. While this allows users to inspect attribute data alongside the network structure, the views are not coordinated, selections or filters applied in one view are not reflected in the other, limiting the analytical value of having both representations simultaneously.

Like Gephi Lite and Cytoscape Web, Polinode treats networks as static snapshots. Comparing network states across time requires loading separate files and manually integrating the differences with no support for interactive temporal analysis or dynamic network visualization.

Polinode is optimized for organizational stakeholders who require actionable insights quickly and with minimal technical overhead. Its structured metric import, basic mapping options, and plain language explanations make it suitable for business audiences. For researchers or analysts who want interactive network exploration, highly customizable visualizations or detailed temporal analysis, Polinode is more restrictive, presenting network metrics as outputs to consume rather than as dynamic dimensions to explore. Like the other platforms reviewed, it offers no dedicated support for simultaneous multi-measure comparison nor dedicated support for temporal networks.

2.6 Evaluation Methods in Visualization

Evaluating visualization systems requires methods that differ from traditional software testing or quantitative performance evaluation. Visualization tools often support exploratory, open-ended analytical tasks, where success is measured not only by efficiency or accuracy but also by how well the system supports insight generation and user understanding. Munzner's nested model of visualization design and validation provides a framework for approaching evaluation across multiple levels of abstraction and for the general workflow of the development of the system [44]. The model distinguishes four levels:

1. *Domain problem characterization*: This level examines whether the system addresses the correct real-world problem. Evaluation at this stage involves understanding domain goals, user needs and analytical questions. Methods include expert interviews, task analysis and surveys, which help verify that the visualization addresses meaningful problems for its intended audience.

2. *Data and task abstraction*: At this level, the focus shifts to whether domain problems have been accurately translated into abstract visualization tasks. Evaluation may involve mapping domain concepts to data structures and task models, ensuring that the abstraction correctly translates user intent. Methods include task analysis, domain expert interviews, and validation of data transformations against domain requirements.

3. *Visual encoding and interaction design*: At this level, the concern is whether the chosen visual representations and interaction techniques effectively support the intended tasks. This includes evaluating the clarity, interpretability and efficiency of visual encodings, as well as the usability of the interactive components. Methods may involve controlled experiments measuring task accuracy and completion time, observational studies of users interacting with the system, and use case scenarios showcasing how effective it is.

4. *Algorithm implementation*: The lowest level focuses on the correctness and computational performance of the underlying system. Evaluation ensures that algorithms produce correct results, handle data efficiently and maintain responsiveness. Methods include unit testing, performance profiling and stress testing with large or edge case datasets.

The nested model emphasizes that evaluation at higher levels cannot compensate for issues at lower levels [44]. For example, even if a visual encoding is theoretically effective, it cannot support insight if the underlying data abstraction is flawed. Conversely, a perfectly implemented algorithm offers little value if it does not align with domain tasks or support meaningful user interaction.

Building on this perspective, visualization evaluation encompasses a broad range of methods, each suited to different evaluation goals. Quantitative performance based evaluations

such as controlled experiments measuring task completion time or accuracy, are appropriate for assessing specific, well defined tasks. However, many visualization systems are designed to support exploratory analysis, where tasks are not always well defined and evolve during interaction. For such systems, qualitative evaluation methods play a central role. These include case studies, use case scenarios, observational studies and user studies which focus on usability, analytical workflows and design reflections. Lam et al. [45] provide a comprehensive taxonomy of visualization evaluation scenarios, highlighting that qualitative approaches are particularly common and appropriate when evaluating visualization systems that support open-ended analytical processes.

In the case of user studies, Lewis [46] concludes that think aloud protocols are commonly employed to gain insight into users reasoning, expectations and decision making. By asking participants to verbalize their thoughts during task execution, researchers can observe how features are discovered, how visual encodings are interpreted and where confusion arises. Furthermore, user studies that focus on usability are shown by Virzi [47] to work well with small numbers of participants, as even a limited sample can reveal a substantial proportion of usability issues. The goal is not statistical generalization, but the identification of recurring patterns, breakdowns and design insights that inform further system development.

Chapter 3
Method

This chapter outlines the methodological approach used to design, implement and evaluate ViNCent 2.0. The methodology follows an iterative design-implementation-evaluation workflow organized into three distinct phases: Design, Development and Evaluation. Together, these three phases address the research questions regarding the improved analysis workflow (RQ1), multivariate attribute support (RQ2), temporal network representation (RQ3), and overall system effectiveness (RQ4). Figure 3.1 provides an overview of the development timeline showing the progression from literature review through design, three-stage development, formative user study, post study refinements and a final evaluation through a use case scenario.

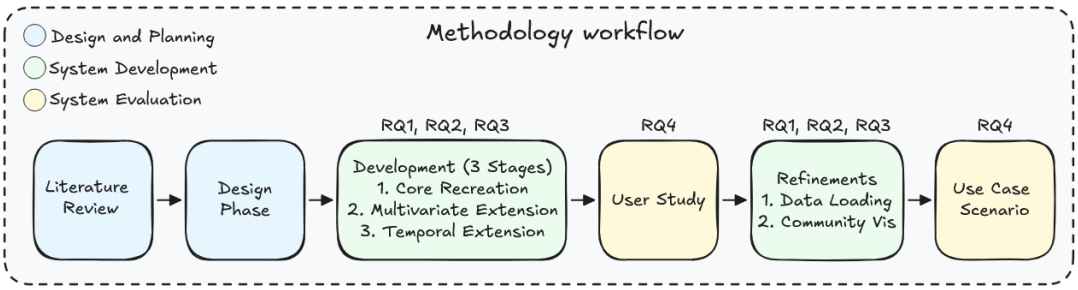


Figure 3.1: Overview of the ViNCent 2.0 development timeline

3.1 Design

This phase establishes the foundation for the entire project by identifying what the system must accomplish and how it should be structured. Building on the literature review and analysis of the original ViNCent and related tools presented in Chapter 2, this phase consists of two main activities: task analysis with system requirements and architectural approach design.

3.1.1 Task Analysis

The task analysis translates theoretical foundations established in Chapter 2 into concrete analytical requirements that guide system design. This analysis operates at the domain problem characterization and data/task abstraction levels of Munzner's nested model (Section 2.6), ensuring that implementation decisions address well defined analytical needs rather than arbitrary feature sets.

Building on the task taxonomies reviewed in Section 2.2, particularly the frameworks proposed by Munzner [5], Lee et al. [13], and Pretorius et al. [16], this analysis identifies specific analytical requirements for centrality focused, multivariate and temporal network

exploration. Rather than designing in isolation, we ground these requirements in limitations of existing tools and theoretical insights about multivariate and temporal networks.

3.1.1.1 Domain-Level Analytical Goals

From the literature review and examination of existing network visualization tools, we identify two primary analytical goals that ViNCent 2.0 must support:

Goal 1: Compare structural importance across multiple dimensions

As discussed in Section 2.2.2, network analysts frequently need to assess node importance according to different centrality measures as they capture distinct structural properties. Furthermore, real-world networks contain domain specific metadata beyond structural measures. As discussed in Section 2.4.1, analysts need to examine relationships between structural importance and these attributes. This creates a domain-level challenge: analysts must compare multiple measures simultaneously to understand the complete importance profile of nodes. As discussed in Section 2.5, existing web-based tools support attribute data but inadequately support multi-measure comparison, as the visual mapping systems only support one or just a few measures at a time through manual channel assignment. On the other side, the original ViNCent focuses exclusively on multi-centrality comparison, as it does not support arbitrary attribute data like categorical labels.

Goal 2: Track how importance and attributes evolve over time

As discussed in Section 2.4.2, many real world networks are temporal, with relationships appearing, disappearing, or changing strength over time. Analysts must track how individual nodes gain or lose importance, identify periods of significant structural change, and understand how topological evolution relates to attribute changes. This creates a domain-level challenge: temporal network analysis requires both understanding trends across snapshots and formulating explanations for observed differences [16]. As discussed in Section 2.5 and Section 2.3, neither existing web-based tools like Gephi Lite and Cytoscape Web nor the original ViNCent provide temporal support. Instead, the tools treat temporal networks as separate static snapshots without integrated navigation, change tracking features or mechanisms for preserving spatial consistency across time periods.

3.1.1.2 Analytical Tasks

These domain-level goals turn into specific analytical tasks that the system must support. We organize tasks by goal and identify which research questions they address.

T1: Compare and rank nodes across multiple measures simultaneously (RQ2)

Analysts must see how nodes rank according to different centrality measures and attributes, identify where rankings agree or diverge and locate nodes with conflicting importance profiles. This task combines Lee et al.'s *attribute-based comparison* tasks [13] with Pretorius et al. *compare* operation [16], applied across both structural and domain specific dimensions.

T2: Filter and focus on node subsets based on attribute criteria (RQ2)

Analysts need to dynamically constrain the visible network to nodes meeting specific criteria and understand how attribute values are distributed across the network. This corresponds to Pretorius et al.'s *identify entities with specific properties* operation [16] and their *understanding estimation tasks* where analysts form approximate characterizations of distributions through visual inspection.

T3: Compare network states and track changes across timesteps (RQ3)

Analysts must understand how node importance, connectivity patterns and attribute distributions evolve over time. This is a core *comparison estimation task* identified by Pretorius et al. for temporal networks [16].

T4: Identify patterns and trends across the network timeline (RQ3)

Analysts must identify trends in how the network evolves over time, detecting gradual developments such as increasing centralization, shifting community structures or growing connectivity, as well as abrupt transitions like sudden expansion phases or the emergence of new hubs. This combines Pretorius et al.'s *understanding estimation task* applied temporally with their *formulating causal explanations* task [16], requiring pattern recognition and hypothesis formation about change drivers through the network timeline.

3.1.1.3 System Requirements

These analytical tasks, together with the original ViNCent specification and functionality identified in Section 2.3, directly inform specific system requirements identified during the design phase. While the specific implementation solutions emerged iteratively during development, the functional requirements themselves guided design decisions from the start. The following mappings connect tasks to requirements and indicate where corresponding implementations are described in Chapter 4.

Requirements for simultaneous multi-measure comparison (T1)

To enable simultaneous comparison of multiple measures per node, the system must provide visual encodings that display all values together rather than requiring users to switch between separate views or visual mappings. Users must be able to immediately see the complete importance profile of any node and visually compare profiles across all nodes. The visual encoding must accommodate both centrality measures and arbitrary node attributes within a unified framework, using consistent visual mappings regardless of attribute type. This unified treatment is essential for examining correlations between all attribute dimensions. For instance, determining whether certain categorical groups tend to have high centrality values, or whether numerical attributes represent structural importance.

These requirements are addressed through the radial bar encoding presented in Section 4.3.1.2 and the type-based color encoding approach described in Section 4.2.

Requirements for supported pattern recognition through reordering (T1)

To reveal patterns and correlations in multi-measure comparison, the system must allow users to dynamically reorganize the network view based on any selected measure or attribute. When users reorder by a centrality measure, nodes with similar structural importance should appear in proximity; when users reorder by a categorical attribute, nodes sharing the same category should group together. This reordering capability must update interactively, allowing users to cycle through different orderings to test hypotheses about correlations or identify divergences where different measures rank nodes differently.

This requirement is addressed through the dynamic node reordering feature presented in Section 4.4.3.

Requirements for attribute type-based filtering (T2)

The system must provide filtering mechanisms tailored to attribute type. For numerical attributes, users must be able to specify value ranges and exclude specific intervals. For categorical attributes, users must be able to select specific categories to include or exclude. These filtering features must provide visual feedback about the underlying data distribution,

showing whether values are uniformly distributed, skewed or clustered, to support informed filtering decisions and understanding estimation tasks. Filters must apply consistently across all coordinated views, ensuring that when a filter is activated, all visualizations update to reflect the constrained node set.

These requirements are addressed through histogram-based filtering for numerical attributes and bar chart filtering for categorical attributes, which is further described in Section 4.4.2.

Requirements for supporting temporal network representations (T3)

The system must represent temporal networks as discrete snapshots where each timestep captures a complete network state, supporting both sequential navigation and direct comparison between distant steps. However, simply storing complete network snapshots for each timestep is inefficient for large networks with many time periods. The representation must therefore balance completeness with efficiency, allowing reconstruction of any timestep state without redundant storage of unchanged elements. This representation must preserve full temporal information, enabling accurate comparison of network states separated by multiple timesteps.

This requirement is addressed through the delta-based representation described in Section 4.1.2.

Requirements for mental map preservation (T3, T4)

To enable reliable tracking of individual nodes across time and support longitudinal analysis of importance changes, the system must maintain spatial consistency during temporal navigation. When users navigate between timesteps, nodes should occupy predictable, consistent positions rather than jumping unpredictably across the display. This spatial stability must be achieved in both diagram views despite their different layout strategies. Additionally, the system should allow users to anticipate where nodes will appear or track where they disappeared, supporting the construction of mental models about network evolution. The layout strategy should ideally encode temporal information visually, for instance, grouping nodes that appeared at similar times, to provide additional analytical value beyond spatial consistency.

These requirements are addressed through foresighted layouts and chronological node ordering which is described in Section 4.5.1.

Requirements for explicit change visualization (T3, T4)

The system must make temporal changes explicit rather than requiring users to mentally compare successive states through memory, to reduce cognitive load. When navigating to a new timestep, users must immediately perceive which nodes are new additions, which nodes have changed attribute values, and which nodes have been removed from the network. This information must be visually encoded directly on the affected elements rather than presented in separate summary statistics. For removals especially, the system must preview which elements will disappear before actually removing them, preventing abrupt visual disruptions that break spatial continuity. Users should be able to observe what is being removed, assess its structural role, and confirm the transition before the removal occurs. This two-stage removal process maintains mental map continuity while supporting the identification of structural change patterns.

These requirements are addressed through visual change indicators and the change preview mode, both of which are described in Section 4.5.2.

3.1.2 Architectural Approach

With the task analysis complete, the design phase transitions naturally into the architectural approach, where requirements are translated into a concrete system structure. To support a web-based system accessible without installation, ViNCent 2.0 adopts a client-server (frontend-backend) architecture separating visualization concerns from computation tasks. This separation of concerns provides several advantages. The frontend can focus on responsive interaction and visual encoding without being burdened by computationally expensive graph algorithms. The backend can leverage scientific computing ecosystems and perform parallel processing when needed. The API-based communication allows the system to scale, as the backend could be deployed on more powerful infrastructure if larger networks require it. The detailed implementation of this architecture, including component interactions and data flow, is presented in Chapter 4, while the following sections motivate the use of frameworks.

3.1.2.1 Frontend Technologies

The frontend is developed using React [48] and TypeScript [49]. React was selected for its component-based architecture, which enables modular and maintainable development. TypeScript was chosen for its static typing to improve code reliability and for reducing runtime errors, which is particularly important when handling complex network data structures.

For visualization, D3.js [50, 51] was chosen for its support of data-driven transformations, scale computations and geometric layout algorithms. D3 provides the mathematical foundations for positioning nodes, computing arc geometries and generating visual scales, while rendering is handled through a combination of SVG and HTML5 Canvas to balance interactivity with performance.

Zustand [52] manages global application state, ensuring that user interactions, filtering operations and temporal navigation remain synchronized across all coordinated views. Zustand was chosen over larger state management libraries like Redux [53] because of its simplicity and smaller scale. The lighter footprint and more straightforward API reduced boilerplate code while still providing the reactive state management needs for coordinated views and data states.

For the user interface, Shadcn [54] components combined with Tailwind CSS [55] provide a consistent and responsive interface. Shadcn was chosen over larger component libraries like Material-UI [56] because of its smaller scale and approach to component ownership. Unlike traditional component libraries that are installed as dependencies, Shadcn copies component code directly into the project codebase. This gives developers direct access to modify the components, allowing for fine-tuned adjustments when a component works almost perfectly but needs small modifications, rather than requiring complete reimplementation or complex overrides.

3.1.2.2 Backend Technologies

The backend is implemented in Python, which was chosen for its large ecosystem of scientific computing libraries, strong community support and suitability for data-driven applications. FastAPI [57] is used as the backend framework due to its high performance and native support for asynchronous request handling, enabling efficient processing of multiple concurrent API requests.

Graph analysis functionality is implemented using the NetworkX library [58, 59] was chosen for its comprehensive collection of graph algorithms and network analysis methods. NetworkX provides built-in implementations for a wide range of centrality measures and community detection algorithms, allowing the system to compute structural properties of graphs without requiring custom algorithmic implementations.

The system supports a broad set of node-level and edge-level centrality measures, including degree-based, eigenvector-based, flow-based and spectral centralities. These measures enable analysis of node importance, information flow, connectivity and influence across both directed and undirected graphs. The supported centrality measures, along with their graph requirements and complexity is summarized in Table 3.1 [60].

Table 3.1: Centrality measures with graph requirements and computational complexity.

Name	**Requirement**	**Time Complexity**
Degree centrality	None	$O(\|V\| + \|E\|)$
In-degree centrality	Directed graphs only	$O(\|V\| + \|E\|)$
Out-degree centrality	Directed graphs only	$O(\|V\| + \|E\|)$
Eigenvector centrality	Connected graphs	$O(k(\|V\| + \|E\|))$
Katz centrality	None	$O(k(\|V\| + \|E\|))$
PageRank	None	$O(k(\|V\| + \|E\|))$
Closeness centrality	None	$O(\|V\|\|E\|)$
Harmonic centrality	None	$O(\|V\|\|E\|)$
Betweenness centrality	None	$O(\|V\|\|E\|)$
Load centrality	None	$O(\|V\|\|E\|)$
Information centrality	Connected undirected graphs only	$O(\|V\|^3)$
Current-flow betweenness	Connected undirected graphs only	$O(\|V\|^3)$
Communicability betweenness	Connected undirected graphs only	$O(\|V\|^3)$
Subgraph centrality	Undirected graphs only	$O(\|V\|^3)$
Laplacian centrality	None	$O(\|V\|^3)$
Second-order centrality	Connected undirected graphs only	$O(\|V\|^3)$
Percolation centrality	None	$O(\|V\|\|E\|)$
VoteRank	None	$O(\|V\| + \|E\|)$
Local reaching centrality	Directed graphs only	$O(\|V\| + \|E\|)$
Trophic level	Directed acyclic graphs only	$O(\|V\| + \|E\|)$
Edge betweenness	None	$O(\|V\|\|E\|)$
Edge load	None	$O(\|V\|\|E\|)$
Edge current-flow betweenness	Connected undirected graphs only	$O(\|V\|^3)$
Edge dispersion	None	$O(\|V\|^2)$
Edge trophic differences	Directed acyclic graphs only	$O(\|V\| + \|E\|)$

In addition, the system supports multiple community detection algorithms, enabling identification of substructures within networks. These algorithms are based on modularity optimization, label propagation, flow-based methods and edge removal strategies. The supported community detection algorithms, their graph requirements and computational complexity is summarized in Table 3.2 [61].

Table 3.2: Community detection algorithms with requirements and computational complexity.

Name	**Requirement**	**Time Complexity**
Louvain	None	$O(\|V\| \log \|V\|)$
Leiden	None	$O(\|V\| + \|E\|)$
Greedy modularity	None	$O(\|V\|^2)$
Label propagation	None	$O(\|V\| + \|E\|)$
Asynchronous LPA	None	$O(\|V\| + \|E\|)$
Girvan–Newman	None	$O(\|V\|\|E\|^2)$
Asynchronous FluidC	Connected graphs only	$O(\|V\| + \|E\|)$
Kernighan–Lin	None	$O(\|V\|^2)$

3.2 Development

The development phase focused on the iterative implementation of ViNCent 2.0. Development progressed in three stages, where each stage introduced new capabilities while maintaining consistency with the original ViNCent's interaction model. The three stages are described in the following sections and provide a high level overview of the implementation process.

The system version produced at the end of Stage 3 served as the basis for the formative user study described in Section 3.3.1. This version included all core analytical features but lacked two supplementary components that were added afterward: the data loading interface which is presented in Section 4.6 and community visualization which is presented in Section 4.7. These refinements, motivated by study findings, are described in detail in Section 5.1.4.

Design decisions were not defined strictly in advance. Rather, while high level requirements were established through task analysis, specific choices regarding visual encodings, interaction mechanisms and interface layout were made during implementation based on internal testing and the formative study. These evaluations guided refinements as the system evolved.

3.2.1 Stage 1: Core System Recreation

The first stage focused on recreating the original ViNCent's core functionality within a new web-based architecture. The goal of this stage was to establish a stable technical and interactional foundation that closely mirrored the behavior of the original system, ensuring continuity with its design principles. Key developments during this stage included:

- Implementing the backend for calculating centrality measures
- Implementing the circular diagram view with radial bar encoding for centrality measures
- Implementing the node-link diagram with force-directed layout
- Implementing the coordinated multiple views framework
- Implementing basic interaction features (hover, selection, zoom, pan)
- Implementing histogram-based filtering for centrality measures

This stage established the foundational user interface layout and interaction model that was preserved through following development. By first recreating the original system, the development ensured that later extensions could be introduced incrementally without disrupting core interactions. Design decisions made during this stage primarily concerned architectural structure and interaction consistency and are described in detail in Chapter 4.

3.2.2 Stage 2: Multivariate Extension

The second stage extended the system to support arbitrary node and edge attributes beyond centrality measures. This stage focused on enabling multivariate analysis while maintaining visual and interactional consistency with the existing views and coordination features established in Stage 1. Key developments included:

- Extending the radial bar encoding to accommodate both centrality measures and node attributes
- Implementing categorical attribute support with a discrete color palette
- Developing bar chart filtering components for categorical data

During this stage, several design decisions emerged from practical implementation challenges, particularly regarding attribute type classification, visual encoding strategies and the coordination of filtering across different attribute types. These decisions were informed by analytical requirements and technical constraints encountered during development and internal testing. The resulting design choices and their rationales are consolidated and discussed in Chapter 4.

3.2.3 Stage 3: Temporal Extension

The third stage introduced support for temporal networks, requiring both new functionality and adaptations of existing components. The primary goal of this stage was to enable exploration of network evolution over time while preserving users' mental models across temporal transitions. Key developments included:

- Updating the backend to support temporal networks
- Implementing discrete timestep navigation with temporal slider controls.
- Developing chronological node ordering and foresighted layouts.
- Adding visual change indicators for both temporal transitions and filtering
- Implementing the change preview feature for removals.

Through this stage, design decisions regarding temporal representation, change tracking and mental map preservation were made in response to analytical goals identified in the task analysis. Rather than being predefined, these decisions emerged as the implications of temporal interaction became apparent through development and testing. The final design solutions are described and motivated in Chapter 4.

3.3 Evaluation

The evaluation phase combines two complementary components conducted at different stages of development. First, a formative user study was conducted near the end of the development phase, after all three development stages (Sections 3.2.1-3.2.3) were complete. This study evaluated a near final version of the system with all core analytical features present. However, it predated two components: the data loading interface (Section 4.6) and community visualization (Section 4.7).

Based on study findings, these major refinements were implemented along with several smaller modifications to existing features. Second, after these refinements were complete, a use case scenario was conducted on the final system to demonstrate analytical capabilities with multivariate and temporal networks using the Delhi Metro dataset. The formative study informed design decisions, which are presented in Section 5.1, while the scenario showcases the refined system's effectiveness in supporting real-world analytical tasks.

3.3.1 Formative User Study

Following the evaluation principles presented in Section 2.6, a qualitative user study was conducted to assess how well the system supports exploratory network analysis tasks. As the visualization is intended to facilitate open-ended analysis and insight rather than optimized task performance, a qualitative approach was chosen over controlled performance-based experiments. In particular, the study focuses on evaluating the data and task abstraction as well as the visual encoding and interaction design levels of Munzner's nested model.

3.3.1.1 Participants

Four participants were recruited, representing similar levels of expertise in network visualization and analysis. Two participants had prior exposure to graph-based visualization through academic coursework but lacked practical experience with network analysis tools. The remaining two participants had minimal to no familiarity with network representations outside of node-link diagrams from online exposure. All participants were computer savvy and possessed at least moderate familiarity with the Star Wars universe, which was used as the domain context for the study dataset.

3.3.1.2 Dataset

The study used the Star Wars character interaction network dataset [62], which represents character relationships across seven episodes. This dataset was selected for several reasons. First, it represents a familiar domain that does not require specialized knowledge, allowing participants to focus on tool usability rather than domain understanding. Second, the dataset exhibits a temporal structure across episodes, enabling evaluation of temporal navigation features. Third, the network is of small size, making it suitable for exploratory analysis without overwhelming participants. Fourth, the characters' relationships create natural analytical questions about protagonists, supporting characters and their importance across the episodes.

3.3.1.3 Procedure

Each session lasted approximately 40–50 minutes and followed a semi-structured interview format based on the think-aloud protocol. The complete interview template is provided in Appendix A. The protocol consisted of:

1. Introduction (2–3 minutes): Brief overview of the study purpose and session structure with emphasis on thinking aloud during exploration.
2. Background questions (5 minutes): Assessment of participants' experience with network visualization, knowledge of centrality measures, and familiarity with the domain (Star Wars).
3. Basic interaction discovery (5 minutes): Participants loaded the Star Wars character network and explored fundamental features including zooming, panning, view switching, node selection, hovering and temporal navigation. This phase served to familiarize participants while allowing observation of which interactions were discovered intuitively.
4. Analytical discovery tasks (15–20 minutes): Participants attempted six open-ended analysis questions designed to assess support for common network analysis tasks, including identifying characters across episodes, finding highly connected characters, detecting bridge characters, observing centrality changes over time and exploring patterns through node reordering.
5. General usability discussion (5–10 minutes): Open-ended feedback on overall experience, most helpful and frustrating features, clarity of visual encodings and suggestions for improvement. Participants also provided usability ratings on a 1–10 scale.
6. Closing (5 minutes): Participants got the chance to ask questions and give closing thoughts on the system.

During each session, interaction with the system was examined and any difficulties encountered were documented. Think-aloud comments were noted to capture how participants discovered and interpreted features, along with the strategies and approaches they used to complete tasks. Moments of confusion and unexpected behavior were also documented and

structured feedback was collected on both specific features and overall usability. The feedback was analyzed qualitatively to identify patterns across participants. The analysis focused on recurring usability issues, features that supported analysis effectively compared to those that caused confusion and concrete suggestions for improving the design.

3.3.2 Use Case Scenario

Following the formative study and subsequent design iterations, the refined system was evaluated through a structured use case scenario using the Delhi Metro network dataset [63]. Scenario-based evaluation is particularly appropriate for exploratory visualization systems like ViNCent 2.0 because it demonstrates analytical capabilities in realistic contexts rather than testing isolated features. The use case scenario adopts an analytical narrative structure in which a representative domain actor, analytical goals and sequential tasks are articulated to demonstrate how system features support exploratory analysis, following established scenario-based design [64] and task abstraction practices in visualization research [5].

3.3.2.1 Dataset Selection

The Delhi Metro network was selected as the evaluation dataset for several reasons. First, it represents a real-world temporal network with clear structural evolution from 2002 (6 stations, 5 connections) to 2019 (260 stations, 269 connections), providing a realistic context for demonstrating temporal analysis capabilities. Second, it contains multivariate metadata that enables demonstration of both centrality-based and attribute-based analysis workflows:

- Opening date (numeric): When each station began operation
- Line membership (categorical, 13 values): Which transit line the station belongs to
- Layout type (categorical, 3 values): Whether the station is elevated, underground or at ground level.
- Interchange status (boolean): Whether the station connects multiple transit lines

Third, the domain is accessible to general audiences without requiring specialized transportation planning knowledge, while still supporting meaningful analytical questions about network growth, hub identification and structural changes over time. Finally, the network size is manageable for detailed exploration within a single evaluation scenario.

3.3.2.2 Scenario Structure

The use case scenario adopts an analytical narrative approach, presenting a realistic analytical workflow from the perspective of a transit network analyst examining the Delhi Metro's evolution. Rather than testing predefined hypotheses or measuring task completion times, the scenario demonstrates how ViNCent 2.0 features support exploratory analysis through connected sequences of analytical tasks.

The scenario is structured around a concrete analytical goal: understanding how the metro network evolved and identifying key stations that serve as major connectivity hubs. This goal motivates a series of representative analysis tasks that illustrate the system's capabilities.

The analyst begins by examining the complete 2019 network state visually prominent stations with high centrality values and observe structural patterns through edge bundling. This overview is followed by investigating categorical attributes like layout type and interchange status, and relating structural properties to domain specific metadata.

The analyst then shifts to temporal navigation, identifying major expansion phases by stepping between timesteps, using change indicators and labels to see when new lines were introduced, and filtering to focus on specific node subsets. To understand how individual

stations evolved, the analyst examines how their structural importance changed across time by combining the circular diagram's centrality encoding with the node-link diagram's topological context. Finally, the analyst tracks station importance across the full temporal extent through dynamic reordering and filtering to identify persistent hubs and stations whose importance increased when new connections formed.

Each analytical step in the scenario demonstrates specific system capabilities while building toward insights about the network's evolution. The workflow shows how users can move between different analytical perspectives using coordinated views, temporal navigation, filtering and interactive features.

The evaluation in Chapter 5 presents the complete analytical workflow including figures showing the system state at key points, descriptions of which features were used and how they supported analysis and the insights derived at each stage. This approach demonstrates not only that the system provides certain capabilities, but how those capabilities combine to support meaningful analysis in practice. The conclusions drawn from this scenario are then summarized in Section 6.3, reflecting on what the demonstration reveals about the system's ability to support exploratory network analysis.

Chapter 4
System Implementation

This chapter describes the complete final design, architecture and implementation of ViNCent 2.0, including refinements made after the formative user study. The system architecture and most core features were present in the version evaluated during the user study. Two features, the data loading interface and community visualizations were added afterward based on study findings, as described in Section 5.1.4. This chapter presents the final system, detailing how rendering and interaction features are realized, and how multivariate and temporal networks are handled. Figure 4.1 demonstrates the full interface, in which the left bar is the settings bar, the center contains the three views, and the right bar contains the filters and a tooltip.

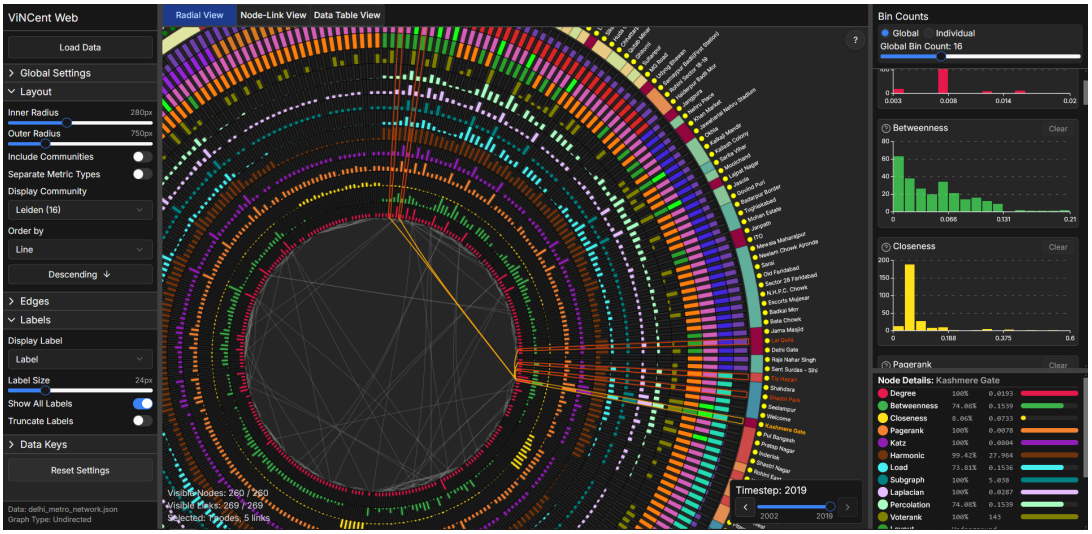


Figure 4.1: Overview of the ViNCent 2.0 interface

4.1 System Architecture

This section describes the complete architecture of ViNCent 2.0 and how data flows through the system during typical usage. Figure 4.2 provides an overview of the system components and their interactions.

4.1.1 Frontend Architecture

The frontend is organized into three distinct layers: state management, data processing and visualization. The first layer is the state management layer. Rather than using a single store, the application employs multiple specialized Zustand stores to better separate concerns. The

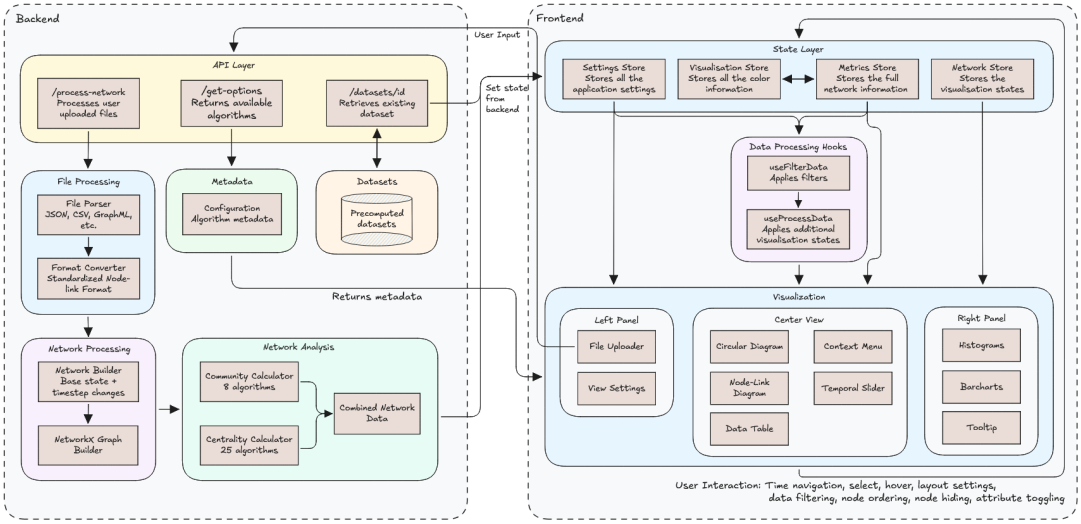


Figure 4.2: Overview of the ViNCent 2.0 system architecture showing frontend and backend components and their interactions.

view settings store manages system wide settings and preferences that control how visualizations are displayed. The visualization store maintains color encoding information and visual mapping configurations. The metrics store stores the network, all metric data and filter states, tracking which measures are active and what filtering constraints have been applied. And finally, the network store maintains visualization specific states including node ordering, hover state, selection state and which nodes are currently hidden. This multi store approach provides clearer separation of responsibility while maintaining the benefits of reactive state management. Each store handles a distinct aspect of the application state, reducing coupling and making the codebase more maintainable.

The second layer is the data processing layer. This layer contains a two-step processing flow in which the data is made ready for visualization. The first step involves applying all the filters to the network data and tracking what changes occur as a result of filtering operations. The second step assigns visual states to data elements, determining which nodes and edges are hidden, removed or pending removal based on settings and user interactions. This is done in two steps to allow for certain settings to not cause recalculations of filters and network changes. This two step process ensures that the visualization layer always receives correctly processed data.

The visualization layer is organized into three primary panels. The left panel contains the data loading component for loading network data and the settings controls that allow users to configure how visualizations are displayed and how interactions behave. The center panel contains the three main visualization views, the temporal slider for navigating through timesteps and the context menu component for additional right-click interactions. The right panel contains the filtering interface including histograms, bar charts and the tooltip that shows detailed information about elements that are hovered. User interactions within these panels, such as selecting nodes, applying filters, or navigating between timesteps, update the relevant Zustand stores, which trigger re-renders that propagate changes through the interface. This unidirectional data flow ensures consistent synchronization across all coordinated views.

4.1.2 Backend Architecture

The backend exposes three primary API endpoints, each serving a distinct purpose in the data loading and processing workflow. The dataset fetching endpoint returns precomputed

example datasets with user-selected metrics. Three example datasets are available: the Les Misérables character network, the Star Wars character interaction network across episodes, and a large synthetic network containing five distinct clusters with attributes. These datasets allow users to immediately explore the system's capabilities without preparing their own data.

The get-options endpoint returns information about all available algorithms and centrality measures. This endpoint provides descriptions, computational complexity estimates, and graph requirement information such as "directed graphs only". This metadata is used for the data loading interface, helping users make informed decisions about which metrics to compute for their network.

The network processing endpoint accepts user uploaded network files and computes requested metrics. This endpoint handles the most complex backend operations, involving multiple processing stages. The first processing stage involves processing and validating the information in the file. The system supports multiple common graph file formats to accommodate the many ways networks can be represented. Supported formats include:

- Edge list representations (TXT, CSV): Source, target, optional weight, optional timestep
- Adjacency list representations (TXT, CSV): Node id followed by connected node ids
- Adjacency matrix representations (TXT, CSV): Full matrix of connections
- Matrix Market format (MTX) [65]: Sparse matrix representation
- GraphML files (GRAPHML) [19]: XML-based graph format
- Custom JSON format: {nodes: [{id, label, timestep, attributes:{...}}, ...], edges: [{source, target, weight, timestep, attributes:{...}}, ...] }

Only edge list, GraphML and the custom JSON format support temporal data. For edge lists, temporal information is edge-based; a node is considered active only when it has at least one active edge. GraphML and JSON formats support explicit node and edge timesteps, allowing finer control over when elements appear and disappear. The backend automatically detects which format an uploaded file uses based on file extension and content analysis, then converts all formats into the internal JSON representation. This normalization step ensures that subsequent processing stages work with a consistent data structure regardless of the original format.

After the first processing step, the system constructs a delta-based representation of temporal networks. Rather than storing complete snapshots at each timestep, the system tracks only the changes that occur between consecutive timesteps. The delta representation consists of a base state that is either empty or contains nodes and edges that do not have timesteps, and a series of delta steps, each encoding only elements added, removed or modified since the previous timestep. This approach significantly reduces the payload size returned by the backend, particularly when networks have many attributes per node but which experience few changes over time, or when most nodes and edges persist across timesteps with only a small number of additions or removals.

The final processing step involves calculating centrality measures and community memberships per timestep, and the final attribute metadata. However, unlike the structural changes, centrality values typically require full recomputation even when only a single edge or node is added or removed, as many centrality measures depend on global graph properties. Therefore, centrality and community data are stored as complete snapshots, full sets of values for each timestep, rather than deltas. This avoids having to deal with precision issues when comparing if changes occur, but at the cost of payload size and memory. The computation modules for centrality and community detection are organized separately, making it straightforward to add new algorithms by implementing them in their respective modules without modifying the core processing pipeline.

After all computations complete, the backend analyzes the full network across all timesteps and constructs the attribute metadata. This metadata remains consistent across all timesteps, ensuring that visual encodings and filtering behaviors do not change unexpectedly as users navigate through time. For each computed measure, the backend determines:

- isCategory: True if the attribute has no more than 15 unique values across all timesteps.
- isBoolean: True if values are restricted to binary representations (1/0, yes/no, true/false).
- isNumeric: True if the values can be interpreted as numbers.
- isCentrality: True if the measure is a centrality metric.
- isCommunity: True if the measure is a community metric.
- Unique values set: All distinct values that appear across the temporal network.
- Range: Minimum and maximum values for numeric attributes.

The boolean flags are not mutually exclusive; an attribute may carry multiple flags simultaneously. A boolean attribute, for instance, is also categorical, while a numeric attribute with a few distinct values may qualify as categorical as well. The frontend uses these flag combinations to determine the most appropriate visual encoding and filtering interface for each attribute, which is further discussed in Section 4.2. The choice of only allowing categorical data up to 15 values might seem arbitrary, but this value was decided upon based on internal testing and constraints in terms of distinct color palettes available.

The backend then finally returns a structured JSON response containing the delta network representation as a base state and delta steps. The centrality and community snapshot measures. The attribute metadata for all attributes and all computed measures. And finally, the timesteps in chronological order.

4.1.3 Complete Data Flow

Typical system usage follows this sequence:

1. *Data Loading*: The user selects a dataset or uploads a file and specifies which metrics to compute. The frontend sends this configuration to the appropriate backend endpoint.
2. *Backend processing*: The backend detects the file format, converts it to the internal representation, constructs the delta network, computes requested metrics, builds attribute metadata and returns the complete response as JSON.
3. *Frontend initialization*: The data is distributed across Zustand stores. Processing hooks activate, D3 generates scales and layout computations execute for both diagram types.
4. *Rendering*: React components render visualizations. The circular diagram draws to Canvas and SVG. The node-link diagram draws to SVG using the force simulation. The data table populates and filter components are rendered and display distributions.
5. *Interaction Loop*: User interactions such as hover, selection, data filtering, node reordering, setting changes, temporal navigation, etc., update stores, triggering re-renders and causing the visualizations to update.

4.2 Color Encoding for Multivariate Attributes

The system applies different color encoding strategies based on attribute type, with all attribute assignments determined by the backend and provided through the attribute metadata. Since flags are not mutually exclusive, an attribute may qualify under multiple types simultaneously. The system therefore follows a priority order when determining color encoding: centrality and community flags take highest priority, followed by boolean, then categorical and finally numeric. This means that an attribute with values such as $1, 2, 3, 4, 5$ that qualifies as both numeric and categorical will be treated as categorical for encoding purposes, receiving a discrete color palette rather than a single color.

For categorical attributes with 10 or fewer unique values, the system assigns discrete color palettes from ColorBrewer [66], ensuring clear visual distinction between categories. When categorical attributes contain between 10 and 15 unique values, the system generates colors by distributing hues evenly across the color wheel using HSL color space (70% saturation, 50% lightness), providing visually distinct colors for the larger number of categories. Boolean attributes, which are categorical, get special treatment. They receive a dedicated two-color palette using light green and pink. While red and green are conventional for true/false, this combination is avoided to account for red-green color blindness.

Numerical attributes are mapped to the sequential *interpolateCool* color scheme from D3. Although this is not optimal since different numerical attributes are independent. However, internal testing showed it is sufficient for a smaller number of numerical attributes.

Centrality metrics, which are numerical, receive special treatment as the number of them is predefined by the backend. They are assigned custom colors inspired by ColorBrewer and the Color Alphabet [67], a set of maximally distinguishable colors. This ensures that centrality measures remain visually distinct from each other and other node attributes.

Community detection algorithms share the *interpolateSpectral* color scale from D3 rather than receiving individual palettes. While this reduces distinguishability when multiple community detection results are displayed simultaneously, it reflects the intended usage pattern where community structures are typically examined one algorithm at a time rather than compared side by side.

Because color assignments are computed over the full temporal extent of the network, the visual encoding remains stable as users navigate through time, allowing reliable interpretation of colors throughout their exploration.

4.3 Central Views

The system provides three primary visualization modes that occupy the central display area, the circular diagram, the node-link diagram and the data table. These views share the same screen space and users can toggle between them based on their analytical needs. Each view offers a distinct perspective on the network data. The circular diagram emphasizes multi-attribute comparison through its radial bar encoding, enabling simultaneous inspection of multiple node attributes. The node-link diagram focuses on the network topology and structural patterns. The data table provides a detailed overview of the full network data in a traditional tabular format. All three views remain synchronized with the global application state, ensuring that filtering, selection and other interactions are consistently reflected across the interface regardless of which view is currently active.

4.3.1 Circular Diagram View

The circular diagram extends the original ViNCent's design to support both centrality measures and arbitrary node attributes within a unified visual representation. This design decision addresses the node measure comparison tasks by enabling analysts to examine both structural metrics and attribute data simultaneously. Following ViNCent's design, each node

is represented as a radial bar positioned along a circular layout. The layout aims to distribute the nodes evenly around the center while making the most of the available space. An overview of the diagram in its base state is presented in Figure 4.3. The following sections describe how the circular layout is computed, how the radial bars encode various measures for each node, and how the edges between nodes are visualized.

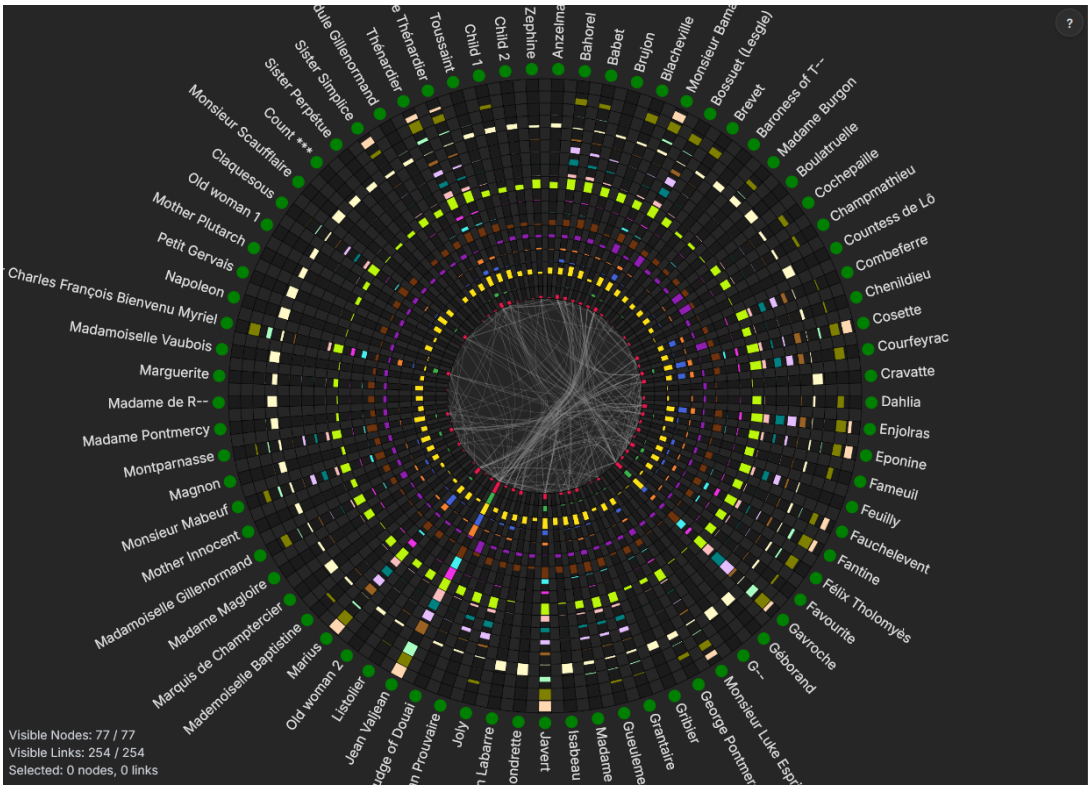


Figure 4.3: Overview of the circular diagram showing nodes represented as stacked radial bars.

4.3.1.1 Circular Arrangement of Nodes

The positioning of nodes is dynamically computed through an angular scale that assigns each node a unique angular position θ_i within the range $[0, 2\pi]$. For a network containing n nodes, each node is positioned according to:

$$\theta_i = \frac{\pi k}{n} + \frac{2\pi i}{n}, i \in \{0, 1, ..., n-1\} \tag{4.1}$$

where $k \in [0, 1]$ represents the gap proportion, controlling the angular spacing between adjacent nodes. This formula divides the circle into n segments, with each segment consisting of a node's angular width $\frac{2\pi(1-k)}{n}$ followed by a gap of size $\frac{2\pi k}{n}$.

The gap proportion adapts to the network size: it is larger for small networks to improve visual separation and decreases progressively as the network grows, allowing larger networks to utilize the available space more efficiently. For networks exceeding $1,000$nodes, gaps are eliminated entirely $(k = 0)$, and nodes occupy the full circumference of the circle.

These angular positions are then converted into Cartesian coordinates for rendering using the standard trigonometric relations:

$$x_i = c_x - rcos(\theta_i) \tag{4.2}$$

$$y_i = c_y + rsin(\theta_i) \tag{4.3}$$

where r is the radius, and (c_x, c_y) defines the center of the diagram. The result is a circular arrangement in which each node occupies an evenly spaced segment, forming the structural basis for the stacked radial bars.

4.3.1.2 Radial Bars

Each node in the circular diagram is visualized as a stacked radial bar, representing multiple measures simultaneously. The bars extend outward from the inner circle, encoding the relative magnitude or category of each metric through color and relative size. The space between the inner and outer radii, denoted as r_{inner} and r_{outer}, defines the available region for drawing the stacked bars. The inner and outer radii are dynamically computed when a network is first loaded, based on both the number of nodes and the number of measures. This ensures that the visualization remains visually balanced and readable across varying data sizes.

The inner radius primarily depends on the number of nodes, ensuring sufficient spacing between node bars around the central circle. For smaller networks, the bars are rendered wider to enhance visibility, while for larger networks, the bar width scales down proportionally to prevent overlaps and maintain clarity. The outer radius is influenced by the total number of measures, with the overall height of each stacked bar adjusting smoothly so that as more measures are introduced, the chart retains a balanced and proportional appearance while trying to retain as much of the layout as possible in view.

Each stacked bar is subdivided into one segment per measure. The position of each segment along the radial axis is determined by a linear scale:

$$y = f(k) = scaleLinear([0, |C|], [r_{inner}, r_{outer}]) \tag{4.4}$$

where $|C|$ denotes the total number of measures currently visible. The linear scale ensures equal spacing between bar segments and automatically reconfigures when metrics are toggled on or off. When both centrality measures and attributes are displayed, an optional visual gap can be introduced between the two groups to make them distinguishable and to improve visual clarity.

For numeric non-categorical data, the bar segments encode the relative magnitude of its corresponding value as a proportion of the full segment height:

$$h(i, c) = \frac{m(i, c) - min_c}{max_c - min_c}(r_{i+1} - r_i) \tag{4.5}$$

where $m(i, c)$ denotes the value of the measure c for node i and min_c, max_c are the global extrema of that measure. This normalized measure ensures consistent scaling and enables direct visual comparison across all nodes for a given measure. In all other cases, the full segment height is used.

4.3.1.3 Edge Representation and Bundling

Edges in the circular diagram can be rendered using one of two selectable modes: simple Bézier curves or Force-Directed Edge Bundling (FDEB) [68]. FDEB was chosen over the original ViNCent bundling approach due to its stronger bundling effect. The original system relied on degree-based control points to create curved edges, but this method produced limited visual groupings. By contrast, FDEB uses an iterative force-based simulation that groups edges with similar trajectories, creating tighter and more coherent bundles that reduce visual clutter while preserving the ability to trace individual connections, at the cost of increased computational complexity.

Users can switch between these modes interactively, with the visualization pipeline recomputing the underlying edge geometry accordingly. In both cases, edges originate from

pairs of angular node positions that are projected into Cartesian coordinates, which serve as the fixed endpoints for all following computations.

In the Bézier mode, edges are represented as quadratic Bézier curves. The control point is computed by contracting the midpoint of the straight line between the two nodes toward the center of the circular layout. This ensures that even edges connecting nodes positioned very close to one another exhibit a curvature rather than collapsing into short and straight line segments. The curvature is independent of the raw edge length and is introduced through radial scaling applied to the midpoint, resulting in consistent and visually coherent curves across all node configurations. An example of the resulting Bézier representation is shown in Figure 4.4.

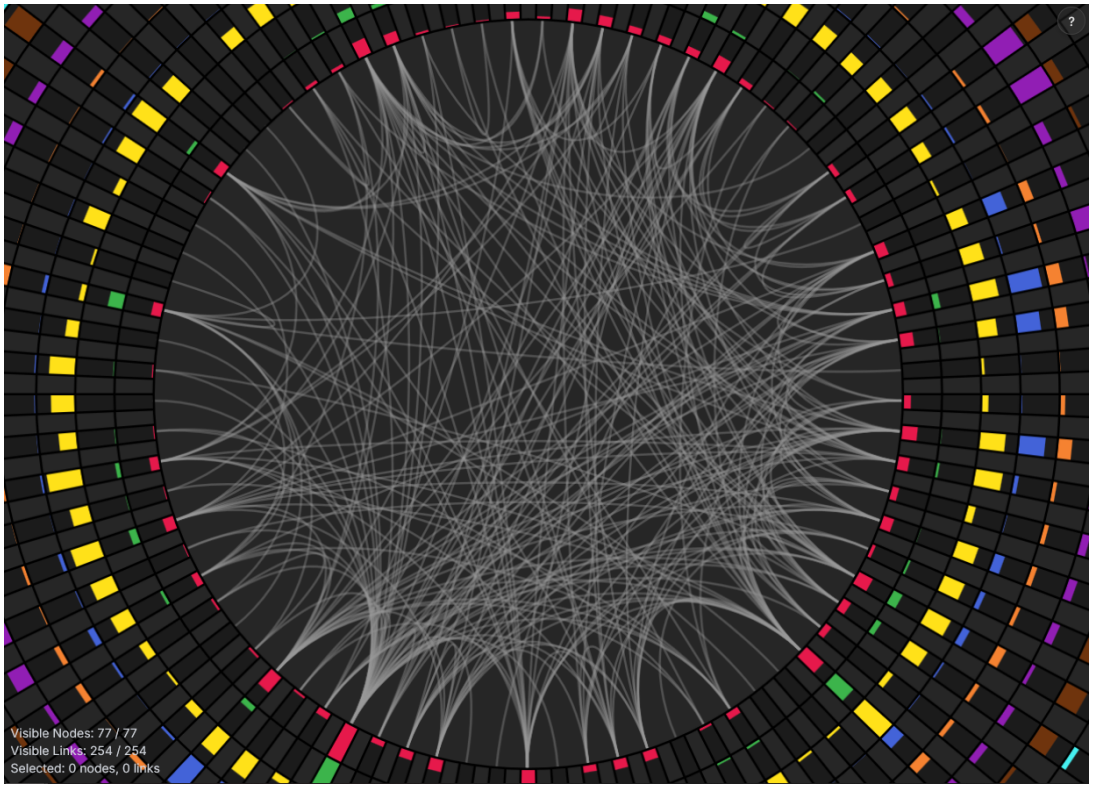


Figure 4.4: Edges visualized as quadratic Bézier curves.

When the FDEB bundling mode is enabled, edges are processed using the FDEB algorithm provided by Holten et al. [68]. In this approach, each edge is subdivided into a polyline of control points that are iteratively updated through multiple cycles of attraction and smoothing forces. During these iterations, angle, scale, position and visibility compatibility measures are computed to determine which edges attract one another. The algorithm first produces a rough bundling using a small number of subdivision points and relatively large update steps, and then progressively refines the result by increasing subdivision density and reducing the step size in later iterations. Figure 4.5 illustrates the resulting bundled structure.

The system allows users to configure key parameters of the FDEB algorithm in order to adapt the bundling behavior to different graph sizes and densities. While several parameters influence the iterative process, two are particularly important. The first is the comparability threshold, which specifies the minimum comparability score required for pairs of edges to exert attractive forces on one another. The second parameter is the initial step size used to displace subdivision points after force computation. This value strongly affects the resulting structure, with small step sizes producing weak or no bundling and large values leading to in-

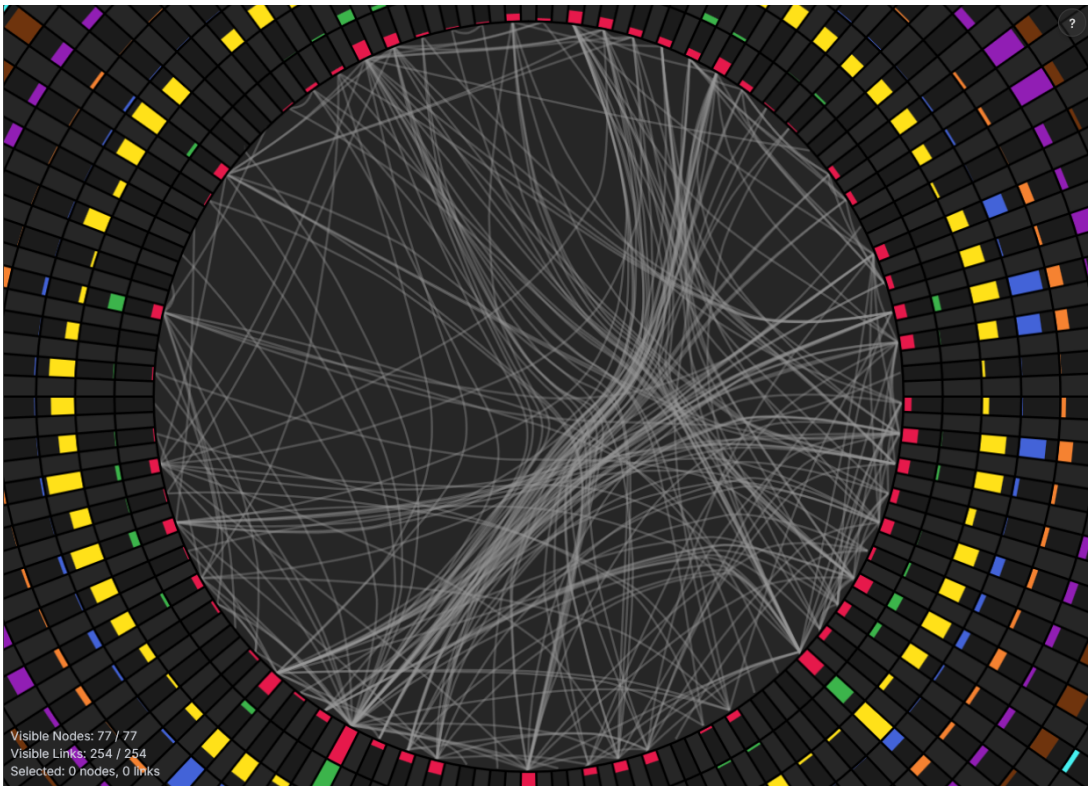
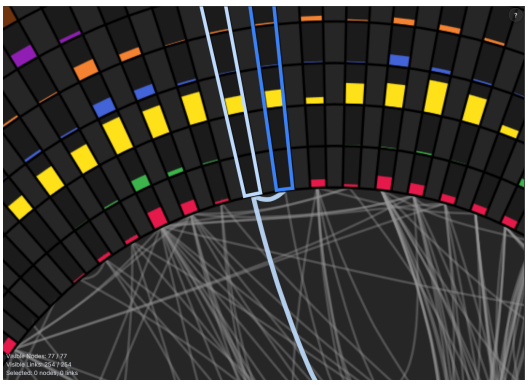


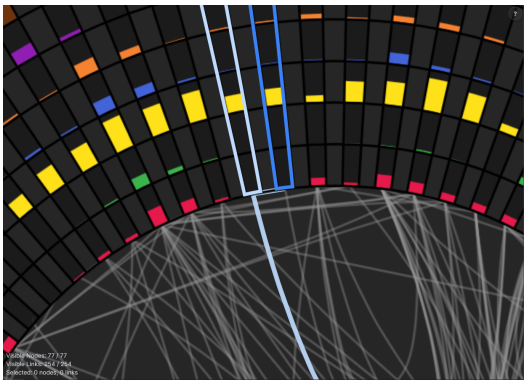
Figure 4.5: Edges visualized with FDEB bundling enabled.

creasingly distorted trajectories. Together, these parameters allow users to balance bundling strength and visual stability within the circular layout.

A small modification is made to the classical FDEB formulation described above because it was originally designed for node-link diagrams. A center-directed force is applied to the first subdivision point of each edge. This force draws the polyline slightly towards the center of the circular layout, ensuring a consistent baseline curvature similar to that found in the Bézier representation but to a smaller degree. The effect of the center-directed force is illustrated in Figure 4.6. The Subfigure 4.6a shows how the added force produces a smooth baseline curvature even for short edges, thus ensuring all edges remain visible. In contrast, Subfigure 4.6b shows the unmodified FDEB behavior where edges connecting adjacent or nearly adjacent nodes tend toward straight local paths, resulting in them being obscured by the border. The center force mitigates this issue while preserving the natural convergence behavior of the remaining subdivision points.



(a) With center-directed force.



(b) Without center-directed force.

Figure 4.6: Comparison of the modified FDEB implementation. (a) Applying a center-directed force to the first subdivision point ensures a smooth baseline curvature. (b) Without this force, edges between adjacent nodes converge to straight trajectories, resulting in them being obscured.

Throughout all bundling cycles, subdivision points are resampled to maintain uniform segment lengths before proceeding to the next refinement stage. After the final iteration, polylines are converted into smooth line paths using D3's *curveBundle* interpolation. For performance, all FDEB computations are executed in a Web Worker, isolating the iterative bundling process from the main rendering thread and ensuring that the interface remains responsive even during rendering for large networks.

4.3.2 Node-Link Diagram View

The node-link diagram uses D3's force-directed layout simulation, which iteratively applies repulsive forces between nodes and attractive forces along links. The simulation gradually reduces movement as the system cools, automatically stopping when the layout reaches a practically stable state. This results in an automatically generated layout in which clusters, gaps and structural patterns become visible. An overview of the base state of the node-link diagram is shown in Figure 4.7.

The node-link diagram supports direct interaction. Nodes can be repositioned by dragging them, which temporarily overrides the simulation forces in order to allow the user to explore alternative layouts or isolate specific parts of the graph. Several force-related parameters can be adjusted, including node repulsion strength and link distance. Modifying these values changes the spread, density and the overall appearance of the layout. Visual properties of nodes and links can also be customized. Node size can be mapped to numeric node attributes. Link thickness can represent edge weights or numeric edge attributes. To reduce visual clutter, labels can be toggled on or off. When active, labels display the corresponding node names directly in the diagram above the node itself.

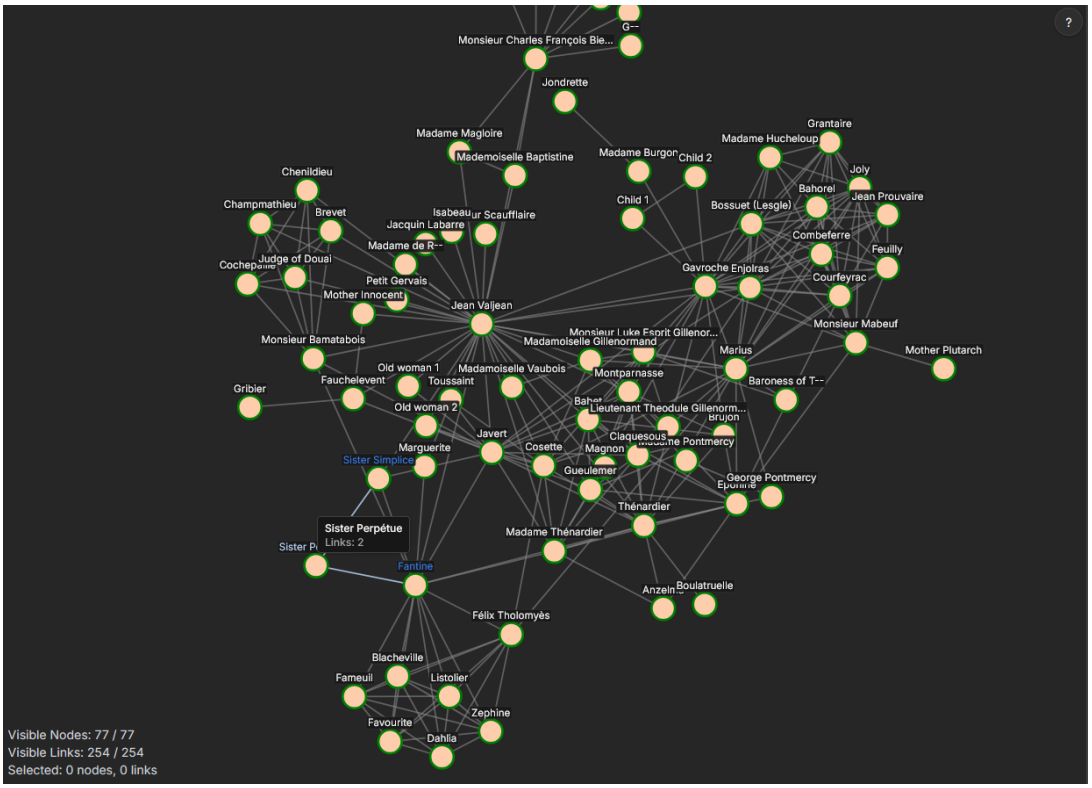
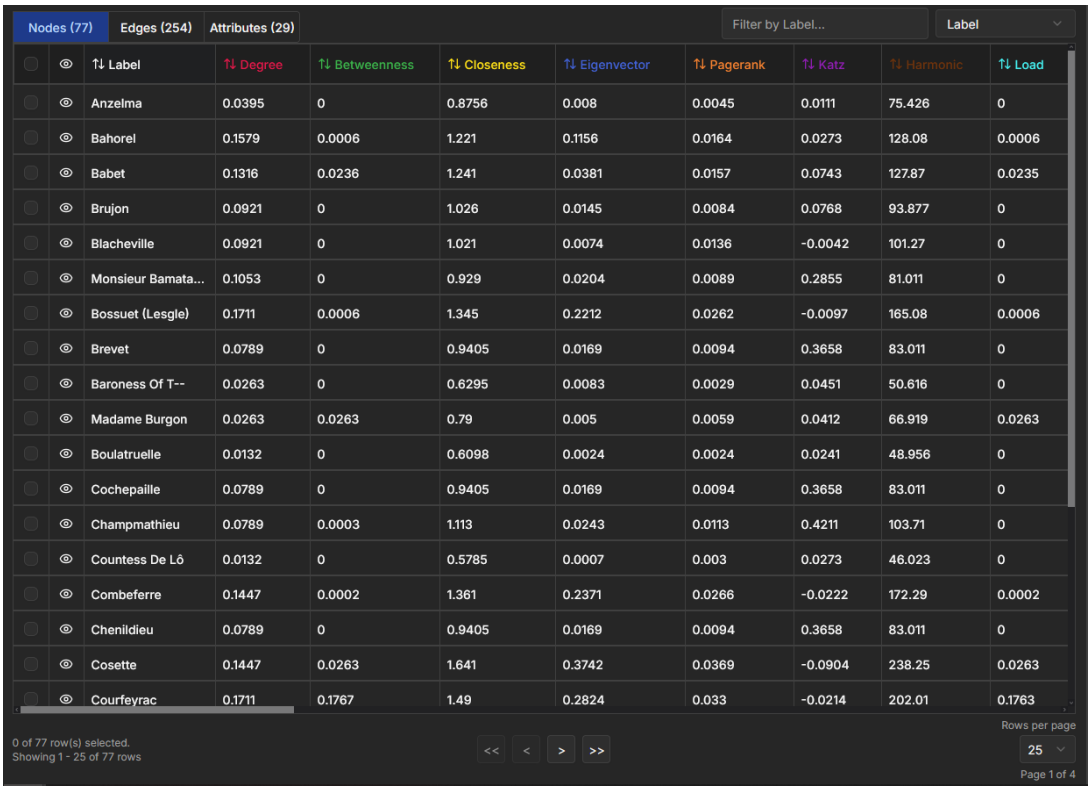


Figure 4.7: Overview of the Node-Link Diagram

4.3.3 Data Table View

To complement the graphical representations of the network, the system includes a tabular data view that lists all nodes and edges together with their associated attributes. In this view, each row represents a single element, while the columns display the different attributes for the given dataset. An overview of the base state of the data table is shown in Figure 4.8.

The data table supports sorting for every column. By clicking a column header, the entries reorder in ascending or descending order, allowing the user to quickly identify nodes with high or low values. The sorting in this view also updates the layout of the circular diagram. A search field enables filtering the table entries that match the given text query. This search affects only the table and does not directly modify the visualizations. To maintain responsiveness for large datasets, the table makes use of pagination and row virtualization. Only a subset of rows is rendered at a time, with additional entries loaded as the user navigates pages or scrolls within the current view. This approach prevents performance issues and ensures smooth interaction, regardless of dataset size. It also contains an attribute table, which shows each attribute as a row, and metadata about them.



Label	Degree	Betweenness	Closeness	Eigenvector	Pagerank	Katz	Harmonic	Load
Anzelma	0.0395	0	0.8756	0.008	0.0045	0.0111	75.426	0
Bahorel	0.1579	0.0006	1.221	0.1156	0.0164	0.0273	128.08	0.0006
Babet	0.1316	0.0236	1.241	0.0381	0.0157	0.0743	127.87	0.0235
Brujon	0.0921	0	1.026	0.0145	0.0084	0.0768	93.877	0
Blacheville	0.0921	0	1.021	0.0074	0.0136	-0.0042	101.27	0
Monsieur Bamata...	0.1053	0	0.929	0.0204	0.0089	0.2855	81.011	0
Bossuet (Lesgle)	0.1711	0.0006	1.345	0.2212	0.0262	-0.0097	165.08	0.0006
Brevet	0.0789	0	0.9405	0.0169	0.0094	0.3658	83.011	0
Baroness Of T--	0.0263	0	0.6295	0.0083	0.0029	0.0451	50.616	0
Madame Burgon	0.0263	0.0263	0.79	0.005	0.0059	0.0412	66.919	0.0263
Boulatruelle	0.0132	0	0.6098	0.0024	0.0024	0.0241	48.956	0
Cochepaille	0.0789	0	0.9405	0.0169	0.0094	0.3658	83.011	0
Champmathieu	0.0789	0.0003	1.113	0.0243	0.0113	0.4211	103.71	0
Countess De Lô	0.0132	0	0.5785	0.0007	0.003	0.0273	46.023	0
Combeferre	0.1447	0.0002	1.361	0.2371	0.0266	-0.0222	172.29	0.0002
Chenildieu	0.0789	0	0.9405	0.0169	0.0094	0.3658	83.011	0
Cosette	0.1447	0.0263	1.641	0.3742	0.0369	-0.0904	238.25	0.0263
Courfeyrac	0.1711	0.1767	1.49	0.2824	0.033	-0.0214	202.01	0.1763

Figure 4.8: Overview of the data table view.

4.4 Interaction Features

Interaction is a core part of how users explore and interpret the relationships displayed across the circular diagram, the node-link diagram and the data table. The system supports quick inspections through hovering and more deliberate exploration through selection and data filtering. Hover and selection interactions consistently highlight both the node of focus and its immediate connections, and the system applies a unified color priority model to ensure that overlapping interaction states remain clear and predictable. The following subsections describe how the main interaction features behave, how they appear in each view and how they relate to one another.

4.4.1 Hover and Select

Hover and select provide two complementary layers of interaction that help users understand both individual nodes and their immediate relationships. Both interactions operate on two levels: the node of focus is emphasized as the primary highlight, while all nodes directly connected to it appear as secondary highlights. This shared structure ensures that users always see the local neighborhood of any node they investigate, whether they are simply moving the cursor or making deliberate selection. Both hover and select always display the labels of affected nodes, regardless of the label visibility setting.

Since hover and select can occur at the same time, the system applies a strict visual priority to avoid ambiguity. A primary selection always takes precedence over all other states, followed by primary hover, then secondary selection and finally secondary hover. This hierarchy ensures that deliberate user action is always the most visually prominent. The combined effect of hover and selection in the circular diagram is shown in Figure 4.9.

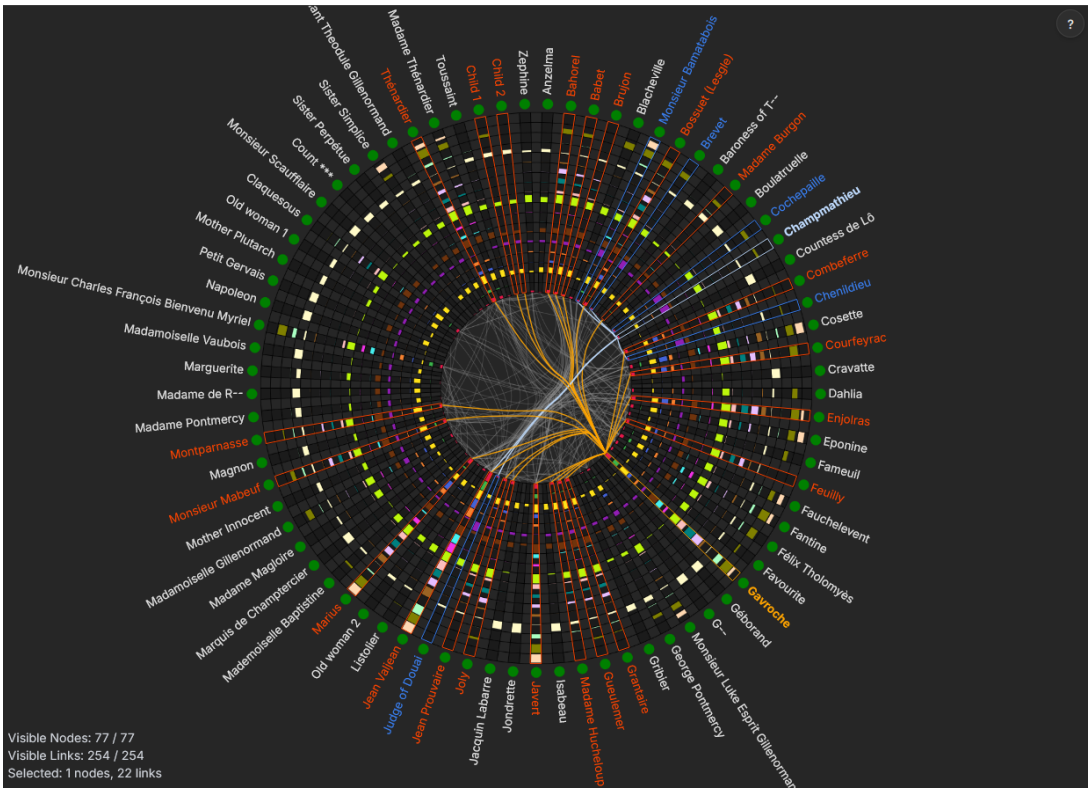


Figure 4.9: Circular diagram with one hovered node and one selected node.

4.4.1.1 Hover Interaction

Hovering offers a quick, lightweight way to inspect a node. Only one node can be hovered at a time, and depending on the chosen settings, the hover effect can either remain visible even after the cursor moves away or disappear immediately when the cursor leaves the interactive area. Hover is displayed in both the circular diagram and the node-link diagram, but each view encodes it differently. In the circular diagram, hovering highlights the border of the associated radial bar, the label of the hovered node and all connecting edges. In the node-link diagram, the label color of the node is modified to indicate hover. The primary hover appears in light blue, and the secondary hover uses a slightly darker light blue color.

4.4.1.2 Select Interaction

Selection functions similarly but represents a more intentional and persistent action. The visual state remains active until the user clears the selection or selects a different node. Unlike hover, the system supports selecting multiple nodes at once. Users can do this either through Ctrl + left click, which adds or removes individual nodes from the current selection, or by making use of region selection, through holding Shift.

The exact region selection functionality depends on the active view. In the circular diagram, the selection area extends along the outer perimeter of the circle, allowing users to drag across node labels to select a group of nodes in a radial sweep. In the node-link diagram, region selection is performed through a freeform lasso tool that lets users trace an arbitrary shape around the desired nodes. In both diagrams, the selection region changes to a lighter gray color when Shift is held to indicate the selection region to the user. In the data table view, multi-selection is supported through the checkbox in the column header, which applies selection only to the rows currently displayed on the active page.

Only primary selections are displayed in the data table view, whereas both primary and secondary selections are shown in the graphical diagrams. Selection is applied consistently across all three views, and its color scheme uses orange for the primary selection and a slightly darker orange for secondary selection, making it visually distinct from hover.

4.4.2 Data Filtering

Data-based filtering complements the hover and selection functionality by allowing users to actively constrain which nodes and edges remain visible across all coordinated views. Whereas hover and selection emphasize relationships between nodes, the filtering interface focuses on their attribute values. To support both numerical and categorical measures, the system implements two different components: histograms for continuous numerical attributes and bar charts for discrete categorical attributes. The two components together form a unified filtering framework in which users can refine the dataset according to any combination of the two attribute dimensions.

4.4.2.1 Histogram Filtering

The histogram is used to represent numerical data by dividing the measure into a fixed number of bins. Users can interact with the distribution in two different ways. The first interaction mode is inclusive range filtering, which is performed by clicking and dragging across the histogram to select a continuous interval of bins. All nodes or edges with values within this interval remain included, while others are excluded. Importantly, this filtering is based on the underlying numeric values, not the bins themselves. As a result, if the data distribution shifts due to different number of bins or changes in the underlying data, the histogram remaps the original value interval using the updated data. This ensures the filter remains stable, consistently reflecting the users intended numeric range even as bin boundaries change during recomputation.

The second interaction mode is exclusive bin filtering, which is performed by right-clicking individual bins to exclude them from the dataset. A bin that is excluded causes all values that fall within that bin to be filtered out while all other values remain visible. In contrast to inclusive range filtering, exclusive bin filtering is explicitly tied to the current bin configuration. If the distribution changes and the system recalculates the bin boundaries, the meaning of each excluded bin changes as well. An excluded region might grow, shrink or shift depending on how the data changes and the filter updates accordingly. This makes exclusive bin filtering a dynamic, bin-based filter that adapts to the current structure of the distribution, whereas inclusive range filtering is a value-based filter that preserves the users' original intent regardless of data shifts. Both filtering modes can be applied simultaneously, as shown in Figure 4.10. The figure demonstrates inclusive range selection and exclusive bin filtering on the degree measure, and shows how this filtering affects the distribution of the closeness measure.

4.4.2.2 Bar chart Filtering

For categorical attributes, the system uses bar charts instead of histograms. Each bar corresponds to a fixed category label, and its height indicates how many nodes or edges belong to that category. Users interact with the bar chart through simple left- and right-click actions. A left click applies inclusive filtering to the category, making only nodes or edges with that categorical value visible while discarding all other categories. A right click applies exclusive filtering to that specific category while keeping all others visible. Figure 4.11 demonstrates both filtering modes for categorical attributes. Because categories are discrete and stable by definition, the meaning of each bar does not depend on the underlying distribution. Even if the number of nodes changes or the dataset evolves, the category labels themselves remain fixed. The system therefore distinguishes between the bin structure of numerical attributes

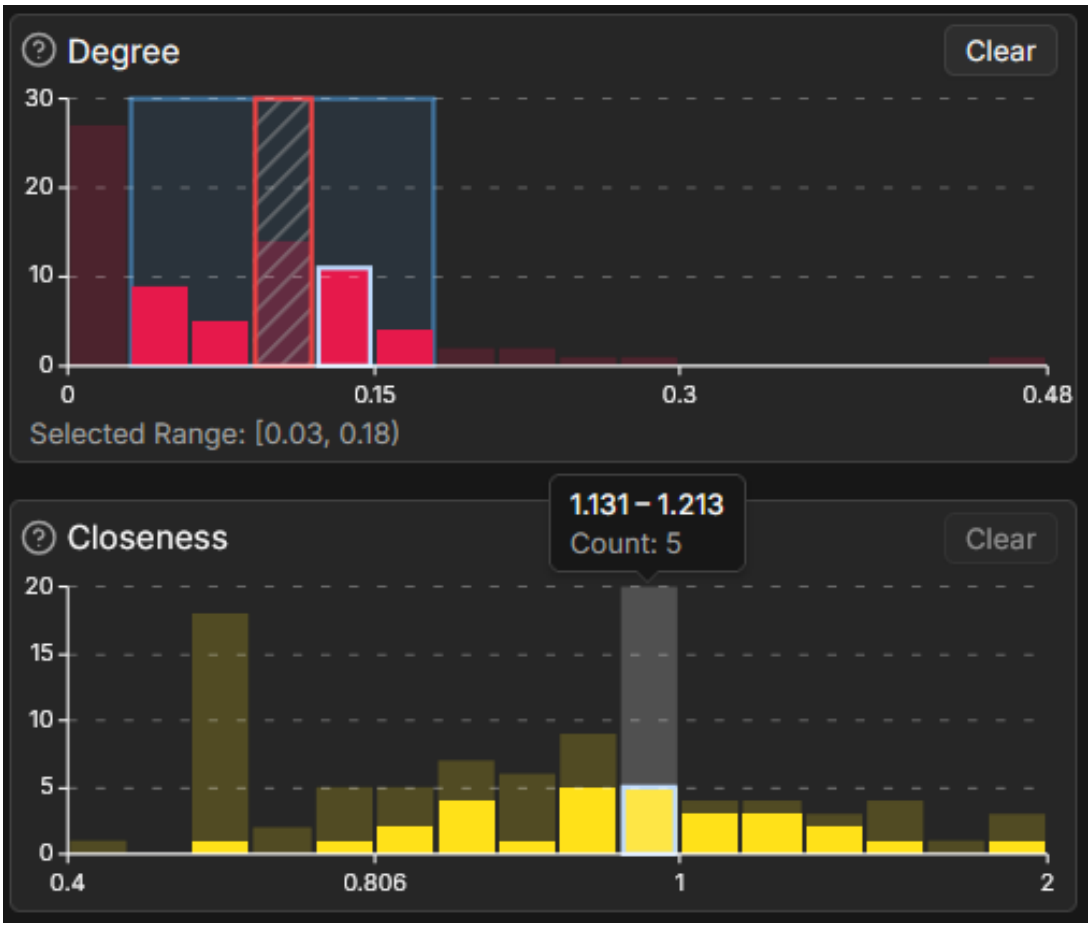


Figure 4.10: Histogram filtering with inclusive range selection and exclusive bin selection applied to the degree measure and the resulting distribution of the closeness measure.

and the static category structure of categorical attributes, ensuring that filtering behaves intuitively in both contexts.

4.4.2.3 Hover Feedback and Filter Propagation

Both histograms and bar charts integrate seamlessly with the hover functionality. When a node is hovered in either the circular diagram or the node-link diagram, the corresponding bin in the histogram or the relevant bar in the bar chart is highlighted. This visual feedback shows the user exactly where the hovered node's value falls within the overall distribution of that measure. Only the primary hover triggers this highlighting, secondary hover and selection highlighting are intentionally excluded from being shown. Limiting the highlighting to the primary hover preserves the clarity and interpretability of the distribution views, avoiding situations where multiple simultaneous highlights would obscure each other and diminish the usefulness of the feedback they provide.

Data filtering in one chart propagates uniformly across all coordinated views, histograms and bar charts as well. When a user defines an inclusive range in a histogram, excludes bins or restricts the dataset to specific categories through the bar chart, the set of active nodes and edges is updated and all three views immediately adjust to reflect this filtered subset. When both histogram and bar chart filters are active, the system applies them in combination: a node or edge must satisfy every applicable filter constraint to remain visible. This allows users to build complex filtering configurations that combine inclusive ranges, exclusive bins and categorical restrictions into a unified workflow.

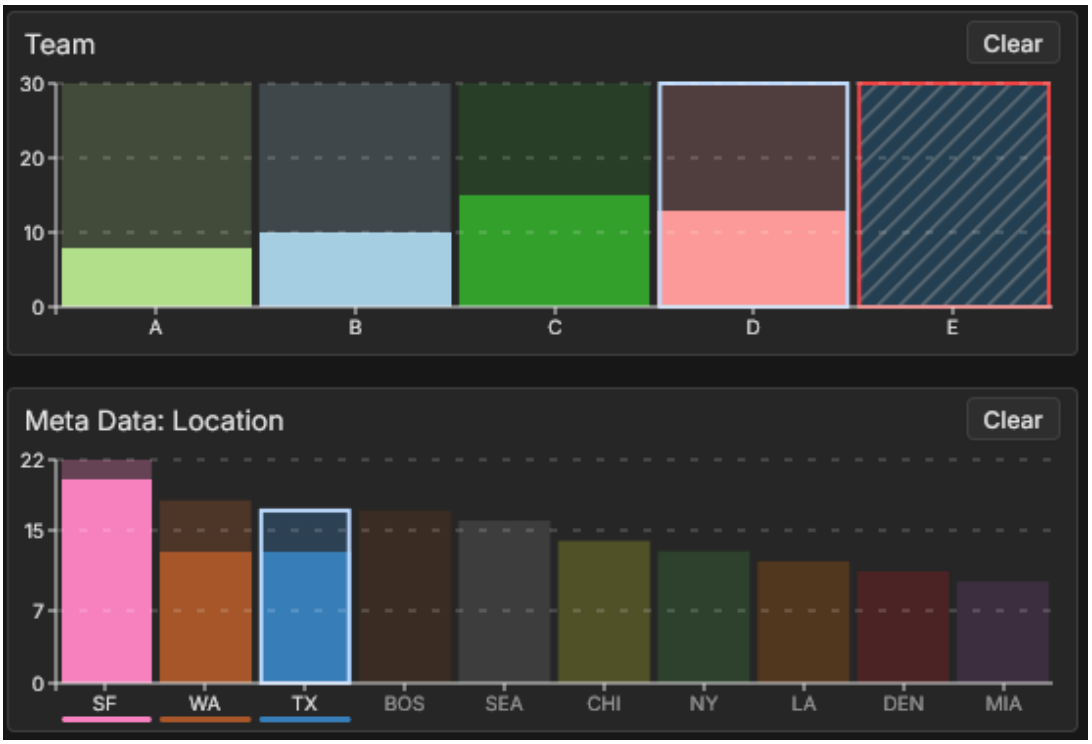


Figure 4.11: Bar chart filtering showing exclusion of a single category and inclusion of three specific categories.

4.4.3 Dynamic Node Reordering

The circular diagram supports dynamic reordering of nodes based on any selected attribute or measure. This interaction allows users to reorganize the circular layout such that nodes with similar values appear in proximity, revealing patterns that may not be visible in the arbitrary initial arrangement. Reordering is performed by selecting an attribute from the settings interface. When a numeric attribute is selected, nodes are sorted by their values in either ascending or descending order, starting at the top of the circle and progressing clockwise around the circumference. When a categorical attribute is selected, nodes are sorted lexicographically by their textual values, resulting in nodes sharing the same category appearing contiguously along the circle. The initial state applies descending order by default, and a toggle control allows users to reverse the sorting direction interactively.

Dynamic reordering allows users to cycle through different attribute orderings to examine correlations, detect outliers or observe whether structural properties align with node metadata. For example, ordering by degree centrality reveals the distribution of hubs in the network, while ordering by a categorical attribute such as node type indicates whether certain types tend to cluster structurally. Reordering on demand provides a flexible way to explore the network from different perspectives.

4.4.4 Other Features

The system includes a number of smaller interaction features that, while less central to the analytical workflow, contribute to a more flexible experience.

4.4.4.1 Label Settings

Users can customize label appearance and behavior through the settings panel. Available options include toggling label visibility, selecting which attribute to display as a label, trun-

cating label length to prevent visual clutter, and adjusting font size for readability. While the selected attribute and truncation settings are shared between both diagram views, font size and visibility settings are maintained independently, allowing users to optimize each view according to its specific layout characteristics.

4.4.4.2 Zoom and Pan

Both diagram views support independent zoom and pan controls, enabling detailed inspection of local structures within each view. To maintain performance during navigation, the system implements automatic label visibility toggling. When users zoom out beyond a threshold where labels become unreadable, the system automatically hides them, reducing rendering overhead and improving responsiveness. Labels reappear when zooming back in to readable scales.

4.4.4.3 Context Menu

All three views implement a right-click context menu that provides quick access to common operations based on the clicked element. Available actions include node reordering selection, attribute visibility toggling, zoom and pan reset, selection clearing and access to the node hiding feature described below.

4.4.4.4 Node Hiding

The system allows users to manually hide individual nodes through the context menu and the data table. Unlike filtering, which retains nodes as empty placeholders in the circular diagram and in the filter distributions, hiding completely removes nodes from the diagram views. Through the context menu, users can hide either a single node or all currently selected nodes. Hidden nodes can be restored either collectively through a "restore all" option in the context menu, or selectively through the data table, which allows users to choose specific nodes to restore. This feature is useful for decluttering visualizations by removing nodes deemed irrelevant to the current analytical focus while preserving the ability to recover them later.

4.5 Temporal Network Visualization

The system handles temporal networks through a discrete snapshot model, where each timestep represents a complete network state. Users navigate between snapshots using a temporal slider component that allows both sequential stepping (forward/backward buttons) and direct jumping to specific timesteps. All the views are synchronized with the current snapshot, ensuring that only the currently visible data is reflected in the visualization.

4.5.1 Layout stability

Both the circular diagram and node-link diagram implement foresighted layout strategies to preserve mental maps during temporal navigation. Inspired by work on preserving mental maps in dynamic graphs [69], both views maintain spatial consistency by considering the full temporal extent of the network rather than computing layouts independently for each timestep.

For the circular diagram, users have the option to display all nodes in the circular diagram, including those not active at the current timestep. Inactive nodes appear as empty radial bars, allowing users to anticipate where nodes will appear or track where they disappeared, while keeping the focus on currently active nodes. To further support temporal comparison and maintain visual consistency, nodes in the circular diagram are ordered chronologically by their first appearance in the network. This creates a visual timeline effect where nodes

that appeared early in the temporal sequence cluster together on one side of the circle, while later additions progress around the circumference. This ordering remains stable across all timesteps, allowing users to identify groups of nodes that appeared at similar times. The combination of foresighted display and chronological ordering ensures that each node always occupies the same angular position regardless of the current timestep unless dynamic reordering is applied.

For the node-link diagram, the force-directed layout is computed for the complete network containing all nodes and edges across all timesteps. Nodes and edges are then selectively hidden or shown based on the current timestep, rather than recomputing the layout for each individual snapshot. This approach prevents layout instability that would otherwise occur if the force simulation were re-run for each timestep. As a result, nodes maintain consistent spatial positions across temporal navigation, preserving users' mental maps even as the visible portion of the network changes.

4.5.2 Change Tracking

To help users track changes across timesteps and filter operations, the system provides visual feedback through change indicators and a change preview feature that work together to make temporal transitions explicit and support informed decision making. This approach is informed by Archambault et al.'s work on difference map readability for dynamic graphs [70].

4.5.2.1 Change Indicators

Visual change indicators appear on nodes in both views to provide an at-a-glance overview of where changes are occurring without requiring mental comparison between snapshots. In the circular diagram, small colored circles appear on the outer edge of each radial bar. Green circles indicate newly added nodes for the current network state, yellow circles indicate nodes whose centrality or attribute values have changed since the previous state, and red circles indicate nodes that existed in the previous state but have now been removed. Hovering a circle shows tooltips; for yellow circles this lists all attribute changes that occurred, while green and red circles simply confirm the node was added or removed. In the node-link diagram, nodes display colored borders carrying the same meaning.

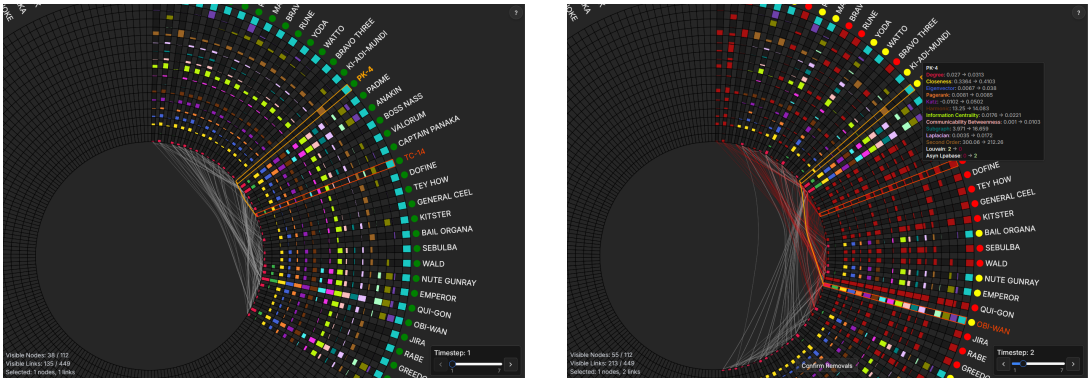
The change indicators are computed relative to the previously viewed network state rather than always relative to the chronologically preceding one. When navigating forward from timestep t to $t+1$, the indicators reflect additions and removals since t. When navigating backward from $t+1$ to t, the indicators reflect the inverse transition: nodes absent in t that were present in $t+1$ appear as removed (red), while nodes present in t but absent in $t+1$ appear as added (green). This behavior ensures that indicators always reflect the transition the user just performed, regardless of navigation direction or jump size. When the timeline slider is used to jump directly between non-adjacent timesteps, indicators correspondingly reflect the full difference between those two states.

4.5.2.2 Change Preview

Recognizing that abrupt removal of nodes and edges can disrupt spatial reasoning, the system implements a change preview feature across both views. When users navigate to a new timestep or apply filters that would remove elements, nodes and edges scheduled for removal are highlighted in red. In the circular diagram, each segment of the radial bar turns red, while in the node-link diagram nodes display red borders. In both views, the edges simply turn red. These highlighted elements remain visible in the visualization, allowing users to observe exactly which elements will be removed and assess the impact on network topology before the change occurs.

Users can confirm the transition by clicking the confirm button that appears at the bottom of the interface, or by continuing their exploration through applying new filters or navigating through time. Only after such confirmation are the marked elements actually removed from the visualization. This two-stage removal process helps maintain spatial continuity and reduces cognitive load during temporal exploration, which is particularly important in the node-link diagram where spatial positioning carries importance.

The Figures 4.12 and 4.13 both show how the layout remains stable while the visible node set changes. The red radial bars and the red-marked nodes and edges are marked for removal as the preview mode is on. Change indicators show which nodes were added, removed or changed between the consecutive steps.



(a) The first timestep. (b) The second timestep.

Figure 4.12: Comparison of two consecutive timesteps in the Circular Diagram.

(a) The first timestep. (b) The second timestep.

Figure 4.13: Comparison of two consecutive timesteps in the Node-Link Diagram.

4.6 Data Loading

The system provides a data loading component that allows users to configure their analysis by selecting a data source and specifying which metrics to calculate. Figure 4.14 presents the view, in which users can either load a precomputed example dataset or upload their own file, and select which metrics to include, which are fetched from the backend. The metrics are organized into logical groups for ease of navigation and conceptual clarity, however this organization is not perfect, as many measures could reasonably belong to more than one group. For example, Node Importance Centralities include measures like degree and VoteRank that rank or indicate the importance of individual nodes, while Distance Flow Centralities include

metrics like betweenness that capture structural properties of the network. Additional groups include community detection algorithms, edge-based centrality measures, and others.

Each metric is accompanied by descriptive metadata to aid users' decision making: a brief description explaining what the metric measures, a note indicating the graph types it is compatible with, for example directed or connected graphs only, and an estimate of its computational complexity. The computational complexity is derived from the time complexity of the underlying algorithms and validated through test runs on larger graphs (4,000 nodes and 8,000 edges). By default, all metrics are enabled, allowing users to quickly proceed with a comprehensive analysis. However, users retain full control to selectively enable or disable individual metrics based on their specific needs.

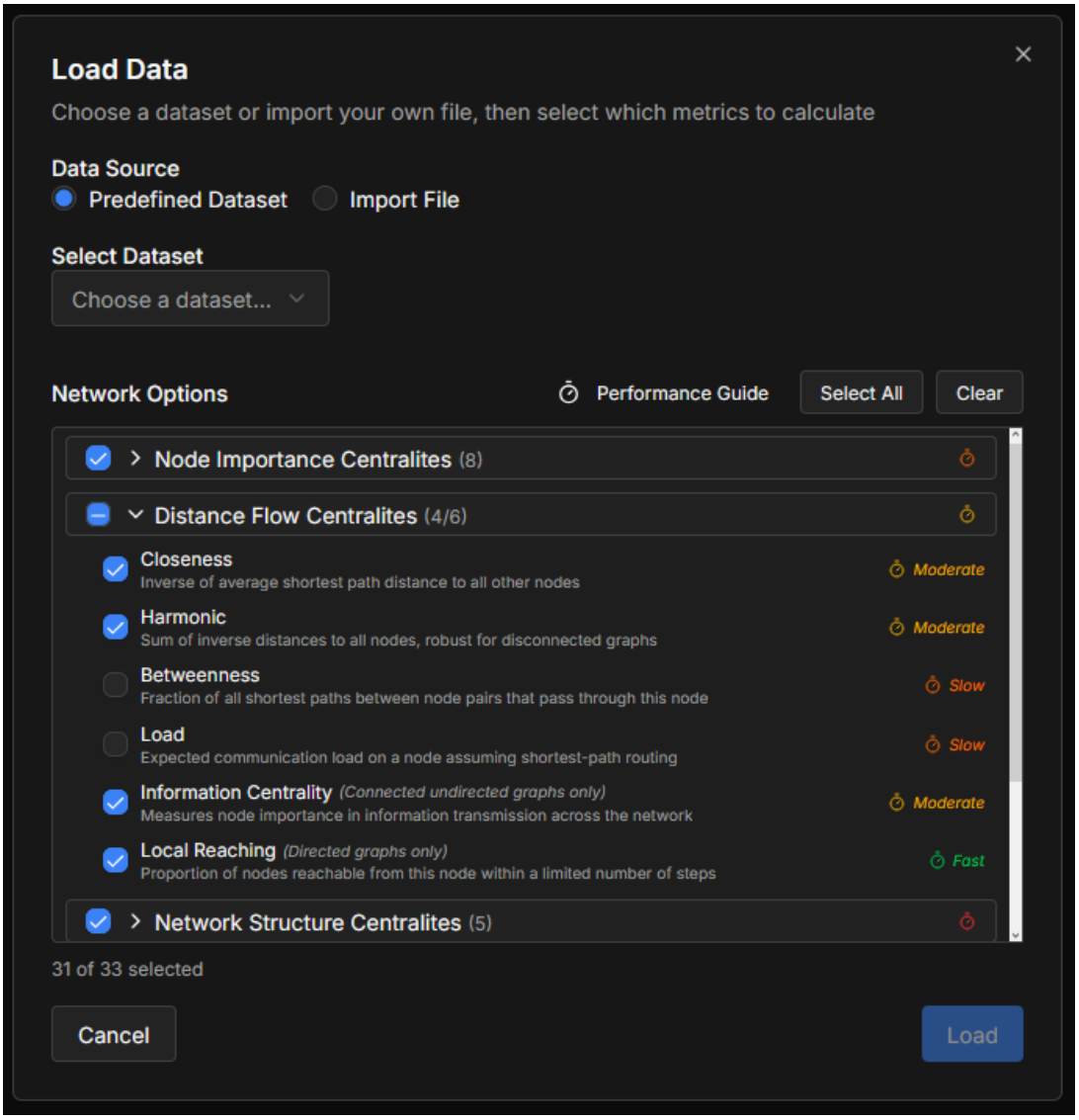


Figure 4.14: Overview of the data loading component which lets users select or upload datasets and pick which metrics to compute.

4.7 Community Visualization

The system implements community detection to identify groups of nodes that are more densely connected to each other than to the rest of the network, as described in Section 2.1.3. This reveals structural patterns such as social groups, functional modules or geographic clusters. The system provides two complementary visual encodings for community membership, which can be used independently or in combination. Users can choose to display community membership either as radial bar segments integrated within the circular diagram's existing attribute encoding, or as a dedicated community view that combines a ring on the circular diagram with consistent coloring in the node-link diagram. Community membership is treated as a categorical node attribute allowing users to also filter based on membership.

When enabled as radial bar segments, each community detection algorithm adds a segment to each node's radial bar using the categorical color encoding from Section 4.2, allowing users to examine community structure alongside centrality measures and other attributes. Users can toggle visibility through the same interface used for other node measures, making it straightforward to compare how different algorithms partition the network by observing color pattern changes across the circular layout.

When a dedicated community view is selected, community membership is encoded both as a ring in the circular diagram and as node coloring in the node-link diagram. In the circular diagram, a ring segment is drawn along the outer boundary of the layout. Nodes belonging to the same community that are positioned adjacently in the circular layout form a single continuous slice, while transitions between different communities are marked by visual separators. In the node-link diagram, nodes are colored according to the same community assignment, providing a consistent representation.

The ring segments extend the hover and selection framework from Section 4.4.1, treating the ring segments as interactive elements. Hovering over a ring segment highlights all member nodes with the assigned color of the community, allowing users to quickly assess a community's size and spatial distribution. However, edges are not highlighted due to performance concerns. Clicking on a community slice selects all nodes that are part of it. The selection state is identical to the one described in Section 4.4.1. Figure 4.15 demonstrates both encoding options enabled simultaneously, with a hovered ring segment highlighting all nodes belonging to that community.

Beyond its visual encoding, community visualization introduces an analytical dimension through its relationship with centrality measures. While community detection identifies groups of nodes that are densely connected internally, centrality measures quantify the structural roles that individual nodes play within and between these groups. The coordinated design of the circular and node-link diagrams enables analysts to examine how highly central nodes are distributed across detected communities and to assess whether specific communities contain disproportionately influential nodes. By exposing community membership as a categorical attribute that can be encoded, filtered and compared alongside centrality measures, the system supports integrated structure-role analysis.

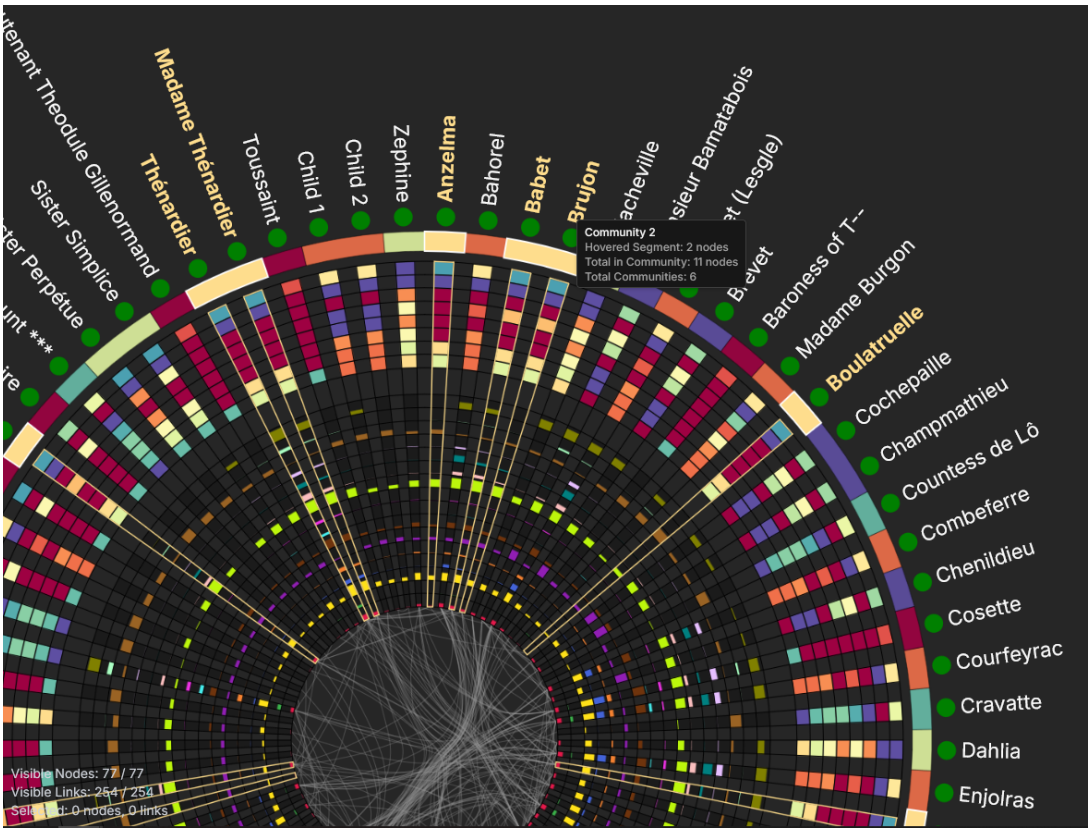


Figure 4.15: Community visualization with both radial bar segments and outer ring encoding enabled.

Chapter 5
Evaluation

This chapter presents the evaluation of the updated ViNCent tool. The evaluation is based on two complementary components: a systematic use case scenario and a small formative user study involving participants. While the use case scenario illustrates the analytical capabilities of the tool in a controlled setting, the user study focuses on usability and user satisfaction. Evaluating both perspectives provides a more comprehensive understanding of the tool's strength and remaining limitations.

5.1 User Study Results

A formative user study was conducted with four participants to assess the usability and analytical support of an near final version of ViNCent 2.0. The complete procedure and participant details are described in Section 3.3.1. This section presents the findings from the study and describes how they informed the following design refinements in the system.

5.1.1 Interaction Discovery

Most basic interactions were discovered quickly and without difficulty. All participants immediately understood zoom and pan controls through standard mouse interactions. View switching was also intuitive, with participants easily finding and using the view toggle controls. Hovering behavior was consistently discovered without prompting, as participants naturally moved their cursor over nodes to explore the visualization. The temporal navigation controls were uniformly understood as well, as participants quickly understood how to step through the episodes.

However, multi-node selection presented a consistent issue, while single-click selection was immediately understood, all participants initially struggled with the multi-selection functionality. They first attempted to build selections through repeated clicking or by looking for a multi-select toggle icon. After receiving a hint about using keyboard modifiers, participants were able to use the feature effectively, but this suggests the need for more explicit visual cues indicating multi-selection capabilities.

5.1.2 Analytical Tasks

Observations from analytical tasks revealed both strengths and limitations of the system. The following task summaries describe how participants approached each task, including some of their reasonings and reflections.

Task 1 (identify characters appearing in multiple episodes): All participants succeeded relatively quickly. They primarily used the node-link diagram combined with temporal navigation, manually checking character presence across episodes. This task demonstrated that the core temporal navigation and visual stability features supported basic comparative tasks.

Task 2 (find highly connected characters in few episodes): This task proved more challenging. Two of the participants attempted to use the filtering interface but expressed confusion. One participant was uncertain about how the filtering interface behaved in relation to temporal changes. The second participant expressed confusion about what exactly they was filtering for. While participants could identify highly connected nodes visually in the node-link diagram, they struggled to systematically combine connectivity patterns with temporal constraints. All four participants ultimately resorted to manual inspection rather than leveraging filtering capabilities effectively.

Task 3 (identify bridge characters between groups): This task highlighted a significant gap in user understanding of centrality measures. When asked to find bridge characters, participants initially did not understand the concept or how it related to network structure. After receiving an explanation that bridge characters connect different groups, participants attempted various approaches including examining filters, examining the circular diagram and guessing based on the connections in the node-link diagram. When prompted about betweenness centrality, participants located high values in the circular diagram, but expressed uncertainty about interpretation: "I'm not sure what these numbers actually mean or if this is right" was one of the comments with others having similar thoughts. This suggests that while the tool provides centrality information, it does not adequately support users in understanding what these measures represent or how to interpret their values.

Task 4 (compare network structure between episodes): In this task, participants generally succeeded but with varying effectiveness. One of the participants used the circular diagram while the others used the node-link diagram. All participants initially used step-by-step temporal navigation, but two found it harder to remember changes when stepping through intermediate episodes. Both then switched to directly clicking on the slider to skip to the target episodes, enabling more effective direct comparison. Notably, only the participant using the circular diagram noticed when new centrality measures appeared in the interface, although did not understand what caused the change.

Task 5 (find characters increasing in importance): This task demonstrated effective use of temporal navigation combined with the circular diagram. Participants stepped through episodes while monitoring individual characters, successfully identifying cases where centrality measures increased. For example, two participants identified C-3PO as appearing in every episode and observed the increasing importance through growing bar segments. This task highlighted the circular diagram's support in enabling users to see multivariate changes over time.

Task 6 (explore patterns through node reordering): Three participants completed this task quickly with some success. One participant took longer, initially scanning for changes in the node-link diagram and filters before switching to the circular diagram. All four participants experimented with different centrality measures to observe how node arrangements changed and to try and draw some conclusions. However, as with the earlier tasks, participants expressed uncertainty about interpreting the significance of differences between different orderings and centrality measures.

5.1.3 Usability Feedback

When asked to rate the overall usability on a 1-10 scale, participants provided ratings of 4, 5, 5 and 5 with a median of 5. Common justifications included acknowledgment that the tool felt usable, but with a steep learning curve. Participants indicated that they gained confidence

as the interview progressed, suggesting that extended use would improve their comfort and effectiveness.

The most frequently mentioned frustration was the lack of explanation for centrality measures. All participants requested tooltips or help text explaining what measures like degree and betweenness represent. Two of the four participants suggested integrating centrality information directly into the node-link diagram, noting that while the circular diagram effectively displayed multiple measures simultaneously, they preferred working primarily in the familiar node-link representation.

Color usage did not generate strong reactions, with most participants noting colors seemed reasonable but that they did not actively attend to color encodings during tasks. This suggests that while the current color scheme is not problematic, it may not be effectively supporting analytical tasks either.

5.1.4 Design Implications

Based on the study findings, several refinements were implemented in the final version of ViNCent 2.0.

The study revealed a gap in users' understanding of centrality measures. While users could locate centrality values in the interface, they lacked the conceptual grounding needed to interpret them meaningfully. To address this, two key features were added. First, a redesigned data loading interface (Section 4.6) was added that provides brief but descriptive explanations of each centrality measure and community algorithm. This allows users to establish conceptual context before exploration begins. Second, explanatory tooltips were added to all centrality histogram labels and community bar chart labels (Section 4.4.2) offering conceptual context on demand.

During the study, participants expressed confusion about how filters behaved in relation to temporal changes. During the study, the inclusive range filter for the histograms was based on the selected bins. This caused confusion when the underlying data distribution changed and the filter range remapped to new values. This was solved by ensuring that the inclusive range filter was based on the selected value range, rather than the bins themselves (Section 4.4.2.1). Additionally, the interface now explicitly displays its bounds as numeric values alongside the visual overlay within the histogram. By keeping the numeric filter bounds fixed and visible, the interface clarifies that only the underlying data distribution changes and not the filter itself.

The study revealed a strong preference for working in the node-link diagram, even when complementary insights were available in the circular diagram. During the study, node and link size encodings were not available in the node-link diagram. This was added to the final system to better support this preference (Section 4.3.2). To improve integration between both diagram views, two new shared visual encodings were introduced. First, configurable node labels allow attributes such as centrality values or categorical information to be displayed directly in both diagrams (Section 4.4.4.1). Additionally, community detection (Section 4.7) was incorporated with consistent visual encoding across both diagram views to better support the identification of groups and clusters within the network. By making community structure explicit in both representations, these encodings allow users to locate and compare groups without switching views. Together, these changes support access to multivariate information in the node-link diagram while reducing the need for frequent view switching.

5.2 Use Case Evaluation

This section evaluates the analytical capabilities of ViNCent 2.0 through a structured use case scenario based on the Delhi Metro network dataset. The evaluation demonstrates how the system supports real-world temporal and multivariate network analysis tasks through a concrete analytical workflow.

5.2.1 Actor

A transit network analyst working for the Delhi Metro Rail Corporation (DMRC) planning department is tasked with understanding how the metro network has evolved from its initial launch in 2002 to its current state in 2019. The analyst has specific goals of identifying key stations that serve as major connectivity hubs, understanding patterns of network growth and expansion, analyzing the relationship between station attributes and structural importance and extracting insights to inform future expansion decisions and capacity planning.

5.2.2 Workflow

The analyst begins by loading the Delhi Metro dataset into ViNCent 2.0 and selects all available metrics. The system automatically computes the metrics and prepares the visualization. Using the temporal slider, the analyst navigates to the complete 2019 snapshot (Figure 5.1) to establish an overall understanding of the network's current state. The radial bars immediately reveal a small number of stations with consistently high values across multiple centrality measures. These stations stand out visually with tall, fully colored segments across all measures, indicating their role as major interchange hubs. The edge bundling pattern shows distinct clusters of densely connected stations, with most edges bundled within temporal groups. Relatively few edges cross the central region of the diagram, suggesting that the network grows through coherent line extensions rather than through distributed connections across the existing infrastructure.

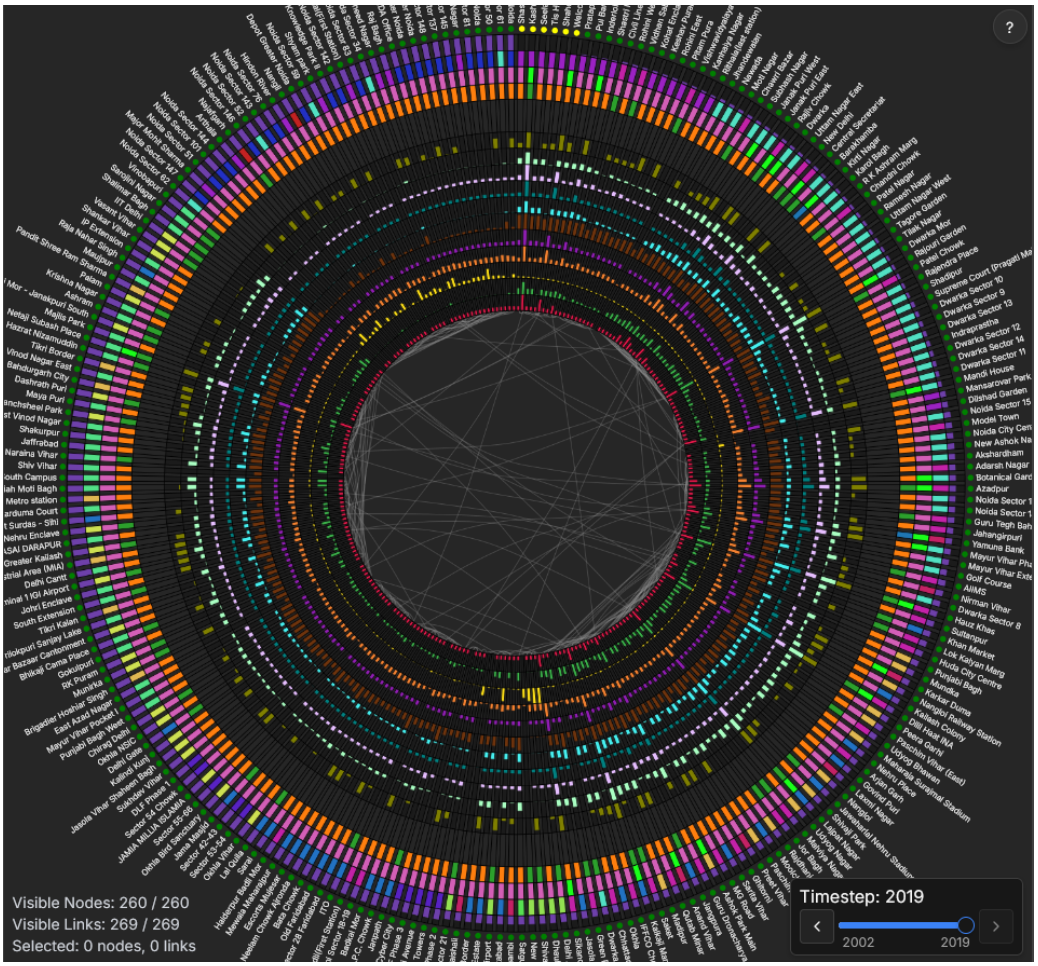


Figure 5.1: Circular diagram visualizing the Delhi Metro network for the 2019 snapshot.

Examining the first categorical attribute ring (layout type) in the circular diagram, the analyst observes that the network is overwhelmingly dominated by elevated stations. Only 2 stations across the entire network are located at ground level, reflecting a deliberate design choice to minimize land acquisition and urban disruption during construction. The second attribute ring (interchange status) reveals that the vast majority of stations are non-interchange stations. Only a small subset serves as transfer hubs, creating a hub and spoke pattern where most passengers must route through specific key stations to change lines. Turning to the attribute bar charts (Figure 5.2), the analyst finds the same distributions represented numerically, reinforcing these observations.

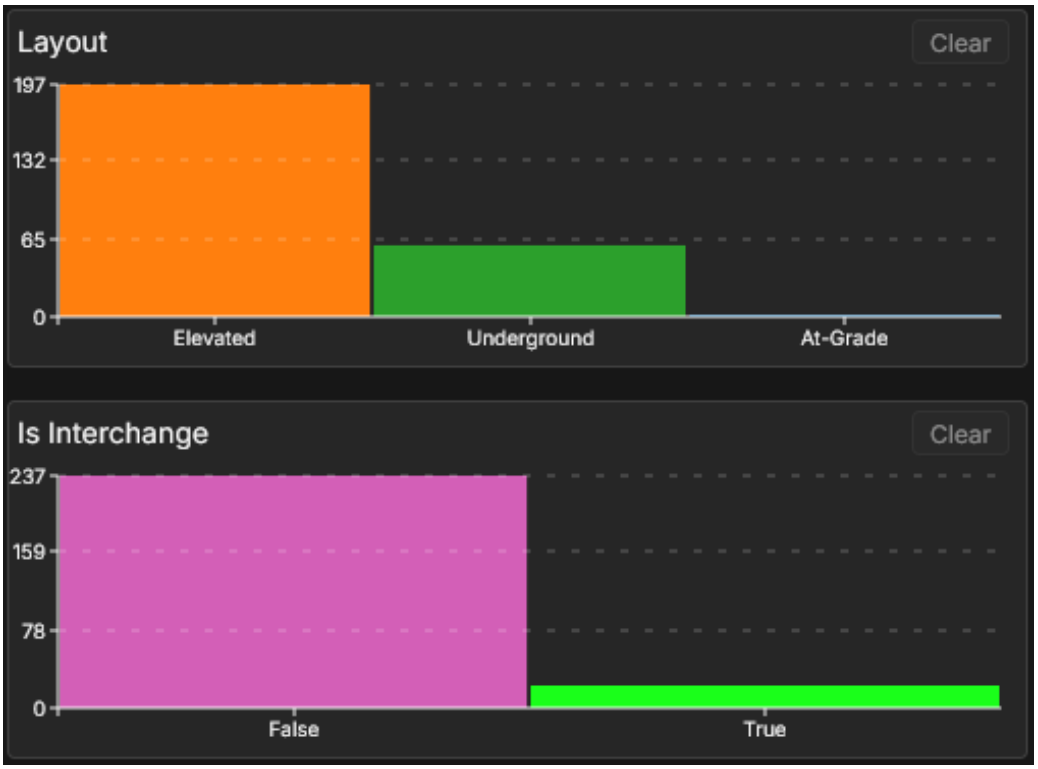


Figure 5.2: Bar charts showing the distribution of layout type and interchange status across all stations in the network.

The analyst then examines the two remaining attribute distributions (Figure 5.3). The line distribution bar chart reveals which colors correspond to which metro lines, a mapping the analyst cross-references with the circular diagram to identify which stations belong to which line. Examining the opening date histogram, the analyst observes distinct waves of construction: the 2002–2003 initial phase with minimal stations, the 2004–2006 first major expansion wave, the 2009–2011 second major expansion wave, and the 2017–2019 recent expansion phase. The years between these waves show minimal additions, reflecting planning and design phases that precede each construction cycle. This pattern indicates that the metro expansion follows deliberate, large-scale planning cycles rather than continuous incremental growth.

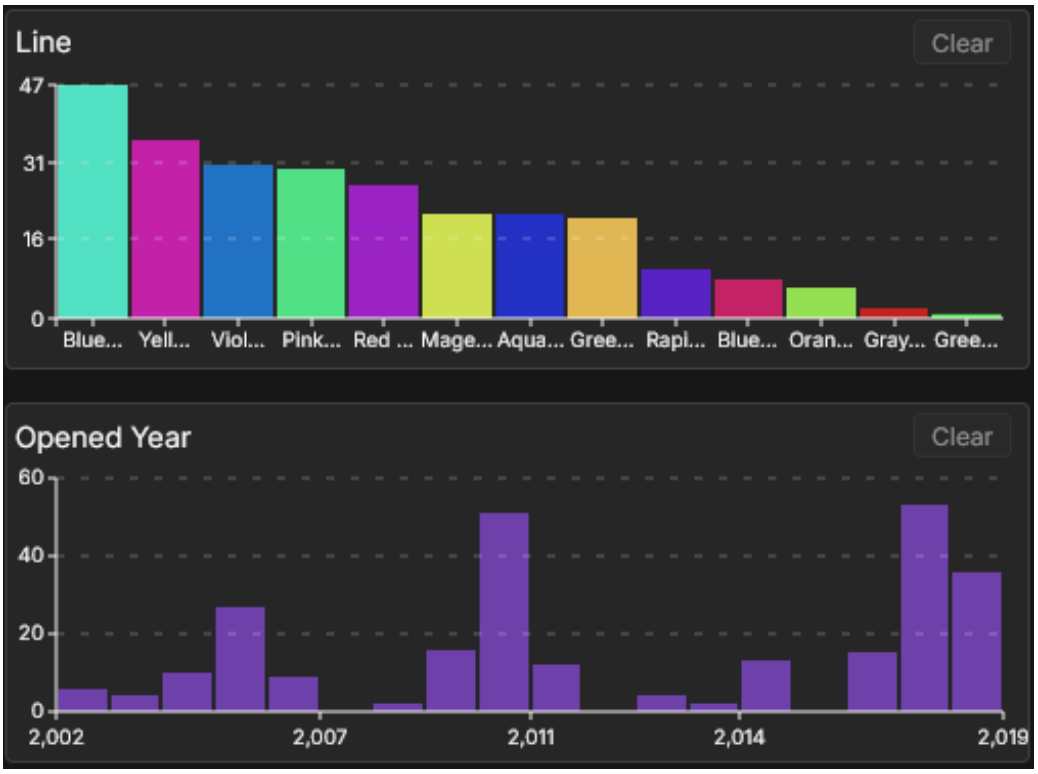


Figure 5.3: Bar chart showing the distribution of metro lines and a histogram of station opening years.

Noticing that 2010 represents one of the largest expansion phases, the analyst decides to examine this period in detail using the temporal slider. By first jumping to the 2009 snapshot and then stepping forward to 2010, the analyst observes which stations are marked with green circles indicating new additions. To better understand line-level changes, the analyst modifies the node labels in settings to display transit line names instead of station names. The circular diagram now reveals clearly that 2010 introduced two entirely new transit lines: the Green Line, a completely new line with multiple stations, and the Violet Line, another new line connecting previously underserved regions, along with a substantial expansion of the existing Yellow Line.

To identify which stations became interchanges in 2010, the analyst first steps back to 2009 and switches to the data table view. There, the analyst searches for "true" in the interchange column and uses the "eye" icon to filter all matching rows, hiding existing interchange stations. Switching back to the circular diagram and stepping forward to the 2010 snapshot, three new interchange stations that were not part of the previous snapshot immediately stand out: one newly created station and two existing stations that have been converted to interchange status. Among the converted stations, one stands out due to its filled centrality bars.

The analyst hovers over the yellow change indicator on this station's radial bar to investigate (Figure 5.4). The tooltip identifies the station as Central Secretariat and reveals substantial increases across multiple centrality measures, indicating an increased importance of the station in the broader network. To understand the topological context, the analyst first restores the hidden nodes and then switches to the node-link diagram (Figure 5.5). The station is highlighted and the spatial layout clearly shows why the centrality measures increased significantly. Central Secretariat now connects the newly introduced Violet Line while simultaneously serving as a transit hub on the substantially expanded Yellow Line. The station has grown from a simple Yellow Line station into a bridge between two major corridors.

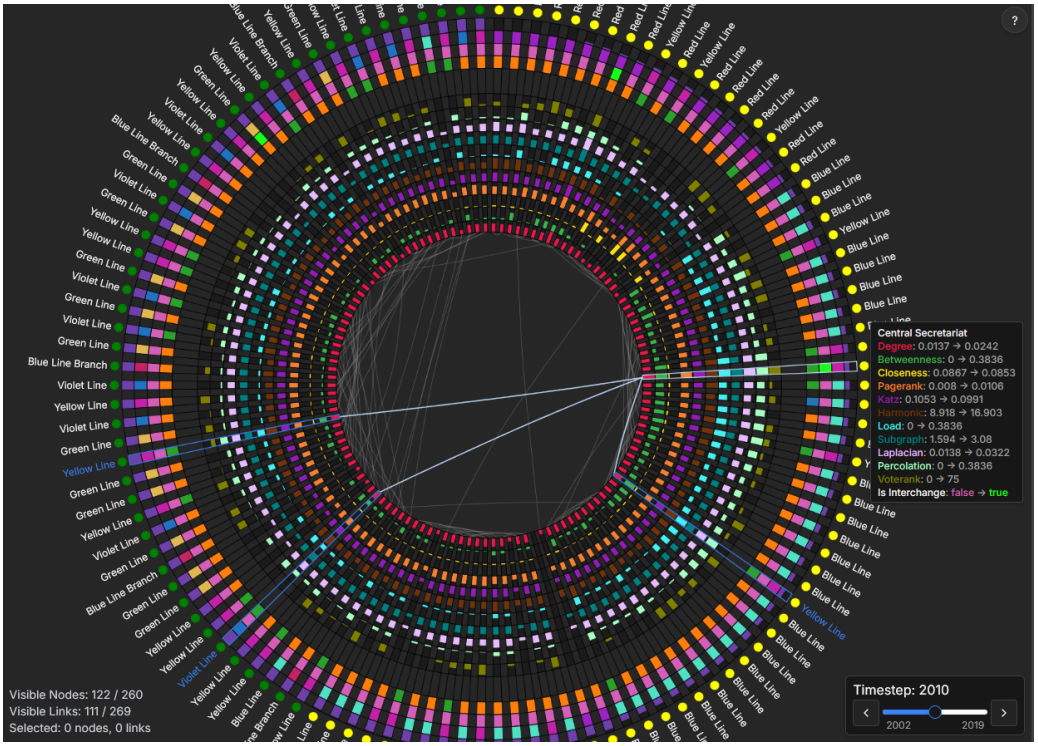


Figure 5.4: Circular diagram visualizing the Delhi Metro network for the 2010 snapshot.

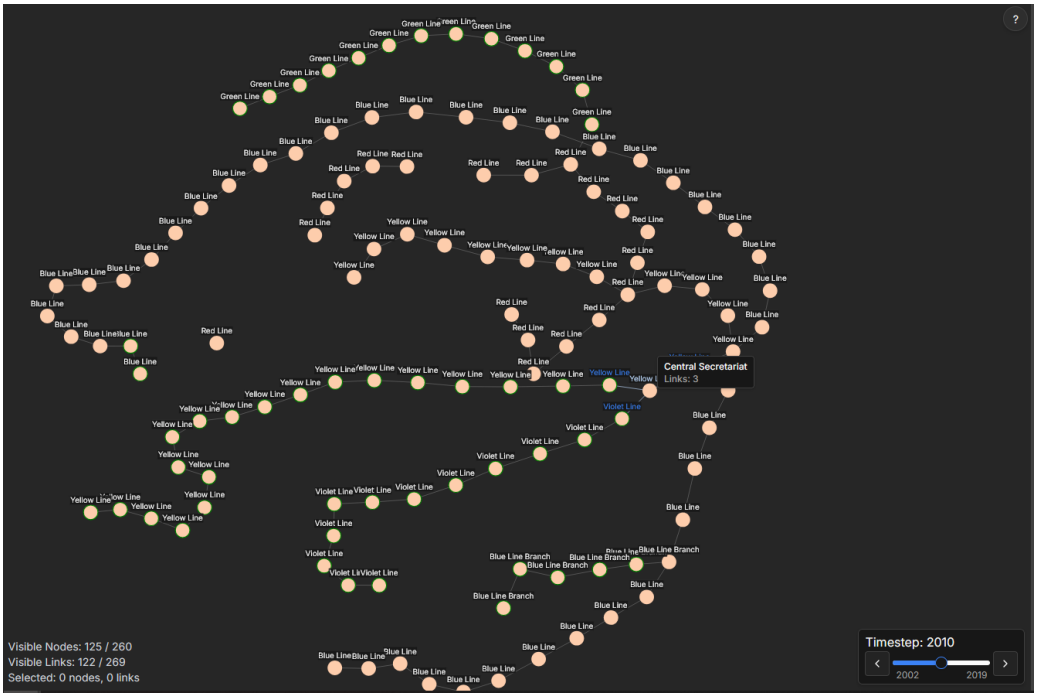


Figure 5.5: Node-link diagram visualizing the Delhi Metro network for the 2010 snapshot.

To track how station importance evolves across the full timeline, the analyst returns to the 2002 snapshot and restores the hidden nodes. The analyst then selects several stations that appear in the initial network. These stations remain selected as the analyst progresses through subsequent timesteps. The analyst uses the circular diagram reordering feature, cycling through different centrality measures at each timestep. This reveals patterns such as stations that were initially central but gradually declined in relative importance as the network expanded around them, stations that maintained consistently high rankings across all measures throughout the timeline, and stations that dramatically increased in importance when new lines connected to them. To focus the analysis more precisely, the analyst applies an inclusive range filter using the histogram filtering interface for betweenness centrality, focusing only on stations with high betweenness scores. With this filter active, the analyst steps through time while maintaining the constraint, observing which new stations remain in the top group throughout multiple expansion phases. Through this longitudinal analysis, certain stations like Welcome emerge as persistent structural hubs: they appear early in the network's history, they maintain high centrality rankings even as hundreds of new stations are added, they consistently score highly across multiple different centrality measures and the node-link diagram confirms they occupy central topological positions.

5.2.3 Insights

Through this systematic exploration using ViNCent 2.0's coordinated views, filtering, temporal navigation and multi-measure encoding, the analyst arrives at several conclusions about the Delhi Metro network.

The network exhibits a clear hub-and-spoke structure, with a small number of key interchange stations serving as essential transfer points. This concentration of connectivity at strategic nodes, rather than distributing it uniformly, is immediately apparent through the radial bar encoding, where a handful of stations consistently stand out across all centrality measures. Growth occurs in discrete waves rather than continuous incremental additions, a pattern revealed through the opening date histogram distribution and confirmed by stepping through the temporal snapshots. Each wave corresponds to the introduction of complete lines or major extensions, reflecting large-scale planning cycles. The overwhelming preference for elevated stations, visible through the layout type attribute ring, indicates a consistent design philosophy to minimize land acquisition and urban disruption.

The longitudinal analysis enabled by temporal navigation and betweenness filtering identifies a small subset of early stations that maintain key centrality positions throughout the full timeline despite massive network expansion. The case of Central Secretariat further illustrates how the system supports causal reasoning by combining change indicators, attribute tooltips and the node-link diagram, the analyst can directly observe why a station's structural importance changed, not just that it did. The concentration of high betweenness at a small number of interchange stations also suggests potential network vulnerability, as disruption at these stations would disproportionately impact system-wide connectivity.

Chapter 6
Discussion

This chapter examines the methodological choices, implementation decisions and evaluation findings from the development of ViNCent 2.0. The discussion addresses the strengths and limitations of each phase, connects findings back to the research questions and reflects on the validity and reliability of the overall approach.

6.1 Method

The thesis adopted a design-implementation-evaluation workflow grounded in Munzner's nested model of visualization design and validation [44]. This approach provided a systematic framework for addressing implementation at multiple levels of abstraction, from domain problem characterization through algorithm implementation.

6.1.1 Design Approach

The task analysis was derived primarily from existing literature rather than through direct engagement with domain experts or target users. While the taxonomies proposed by Lee et al. [13], Munzner [5], and Pretorius et al. [16] provide well-established theoretical foundations, relying exclusively on literature-based task abstraction introduces potential gaps between theoretical requirements and actual user needs.

Alternative approaches to task identification exist. Formative interviews with network analysts in the design phase could validate whether literature-based tasks align with real-world analytical workflows and reveal domain specific requirements that existing taxonomies may not capture. For instance, interviews might uncover whether analysts actually need to compare multiple centrality measures with categorical attributes simultaneously in practice, or whether they typically focus on one or the other at a time.

The lack of early user involvement means that the identified analytical tasks reflect what the literature suggests users should need rather than what they actually do. This creates a risk that the system optimizes for theoretical workflows that do not match realistic usage patterns. Participatory design methods or studies of network analysts represent alternative approaches that provide stronger empirical grounding for task identification, though they require substantially more time and access to domain experts than literature-based analysis.

6.1.2 Development Approach

The three-stage development process followed an incremental strategy: recreating core functionality, extending to multivariate attributes, then adding temporal support. This approach ensured technical stability at each stage and preserved continuity with the original ViNCent design. However, this sequential approach also introduced inefficiency. Design decisions were made reactively during implementation rather than being fully specified in advance. While this flexibility allowed adaptation based on emerging insights, it also required refactoring as later stages revealed technical limitations. For instance, the addition of temporal

networks required refactoring all major components, as new requirements for foresighted layouts, change tracking and the delta-based network representation emerged.

Alternative development approaches include more structured design specification upfront, potentially informed by design workshops or paper prototyping with target users, which could identify requirements earlier and reduced technical debt. The trade-off between iterative flexibility and upfront specification is inherent to system development [71]. The chosen incremental approach prioritized rapid prototyping and continuous refinement over comprehensive early planning, which is appropriate for exploratory visualization systems where requirements may not be fully known in advance.

6.1.3 Evaluation Approach

The evaluation combined a formative user study with a use case scenario to assess both usability and analytical capabilities. This qualitative approach aligns with established practices for exploratory visualization systems, where the goal is to support open-ended insight generation rather than optimize task performance [45]. The evaluation methodology involves inherent trade-offs that affect the interpretation of findings.

The formative user study involved only four participants without network analysis expertise, providing insights into novice interaction patterns rather than expert analytical workflows. This small size is appropriate for identifying major usability issues [47], but limits generalizability and prevents statistical analysis of usability metrics. The lack of domain expertise among participants means the study evaluated conceptual understanding and interface learnability rather than effectiveness for realistic analytical tasks, which is a natural consequence of the formative study design. Participants struggled with network concepts like "bridge characters" and centrality interpretation, revealing the learning curve for novices rather than effectiveness for experienced analysts. Alternative evaluation designs with domain experts would likely reveal different insights focused on analytical efficiency, feature completeness and integration with existing workflows.

Related to this is the timing of the study itself. The formative study was conducted on a near final version of the system, after all three development stages were complete. While this ensured participants interacted with a stable and representative system, it also meant that the window for incorporating feedback into the design was limited. A study conducted earlier, for instance after Stage 1 or Stage 2, could have identified fundamental interaction or interface issues with more time available to address them. The trade-off is that earlier versions would have lacked the multivariate and temporal features central to the research questions, potentially producing feedback of limited relevance to the final system.

The use case scenario provides valuable demonstration of analytical capabilities but reflects a single network type and the developers familiarity with the system. The Delhi Metro network exhibits specific characteristics, steady growth, clear construction phases, small size, simple categorical attributes that do not represent the full range of real-world networks. Networks with different properties would stress the system differently, frequent node/edge removals rather than steady additions, categorical attributes exceeding the 15 value threshold, hundreds of timesteps requiring different navigation strategies, or dense networks where edge bundling becomes computationally expensive. Additionally, the use case scenario was conducted by the system developer with complete knowledge of all features and their intended usage, meaning it does not establish how effectively the system supports real analysts working with their own data and questions.

Alternative evaluation approaches exist that address these limitations. Longitudinal studies with domain experts using their own datasets over multiple sessions [72] would reveal whether the system supports sustained analytical work. Quantitative task-based studies measuring completion time and accuracy for specific analytical questions would quantify efficient gains or overhead. Evaluation with diverse network types would establish general-

izability across domains. Each approach involves trade-offs between validity, experimental control, resource requirements and types of insights generated [45].

The chosen evaluation approach reflects a decision made toward the end of the project, as the conditions required for a summative evaluation did not fully materialize during the thesis. Consequently, the evaluation focuses on formative methods that identify usability improvements and illustrate analytical capabilities through a use case scenario. The results therefore provide formative insights and capability demonstrations rather than summative validation of effectiveness for expert users. However, such qualitative system evaluations are still useful in practice for early stage research prototypes [73].

6.2 Implementation

The implementation phase transformed the methodological foundation into a functioning web-based visualization system. This section examines the technical decisions and architectural choices.

6.2.1 Architectural Decisions

The client-server architecture provided clear separation between visualization and computation, enabling independent development and potential scaling. However, certain architectural choices introduced both benefits and constraints worth examining.

The decision to use multiple specialized Zustand stores rather than a single centralized store improved code organization by separating view settings, metrics data and network interaction state. Each store handles a distinct concern, reducing coupling and making components more maintainable. However, this approach also distributes state across multiple locations, making it more challenging to reason about data flow and potentially introducing synchronization issues if stores are not carefully coordinated. A more complex application might benefit from a formal state management pattern like Redux, despite its additional boilerplate, to enforce stricter data flow.

The backend computes all centrality measures and community detection algorithms on demand when a dataset is loaded. This approach ensures that users always work with fresh calculations based on the input data, but it introduces latency during initial loading, particularly for large networks. Currently, the backend does not cache results. As a consequence, reloading a dataset, regardless of whether the data has changed, triggers a full recomputation of all measures. In addition, the backend lacks request timeouts and workload management features. Loading very large datasets can therefore block the backend process, temporarily preventing it from handling other requests. These limitations were not addressed in the implementation, as the goal of this thesis was not to develop a fully optimized, production ready web platform, but to demonstrate a functional web-based visualization system that enables interactive analysis of temporal and multivariate networks. For a deployed system serving multiple users, a caching layer that stores computed results keyed by network hash and algorithm parameters could significantly improve response times. This is further discussed in Section 7.2.

The system employs delta-based encoding for temporal networks to reduce both payload size and memory usage. Instead of storing complete network snapshots for each timestep, only the incremental changes (deltas) between successive states are stored. Each snapshot is then reconstructed by applying the accumulated deltas to the current network state, allowing efficient storage and transmission while preserving the full temporal structure of the data. However, centrality values are stored as complete snapshots rather than deltas, as centrality measures are derived properties whose changes are difficult to represent meaningfully in incremental form. This results in a hybrid network representation approach that optimizes memory for incremental changes in network structures and attributes, while introducing ad-

ditional conceptual complexity in the data representation. A future refinement might explore alternative network structures that reduce this complexity while maintaining performance.

6.2.2 Missing Features from Original ViNCent

Two features from the original ViNCent were not included in the reimplementation: clustering and data export capabilities. These exclusions reflect prioritization decisions made during development but represent limitations of the current system.

The original ViNCent supported k-means and hierarchical clustering based on centrality profiles, allowing users to group nodes with similar importance. While ViNCent 2.0 includes community detection algorithms, these are fundamentally different, as communities are defined by edge connectivity whereas clustering groups by attribute similarity. This absence directly impacts RQ1, as it removes an analytical capability that the original system provided. The decision not to reimplement the feature was due to time constraints as the first stage of development took longer than expected. Implementing clustering would have required additional backend algorithms, UI components and visualization strategies for displaying cluster membership. The extensions to support multivariate and temporal networks were prioritized over recreating this feature. Including clustering in a future version would complement the existing analytical capabilities, providing a more complete toolkit for exploring both structural and attribute-based patterns in networks.

The original ViNCent allowed users to export analyzed networks in ViNCent GraphML format, preserving computed metrics for future sessions and to export visualizations such as static images. These features were not prioritized in the reimplementation due to time constraints. Future work could extend beyond simple data export, enabling more advanced capabilities such as sharing visualizations via the web. Implementing such extensions would provide a more complete and flexible toolkit for exploring and preserving network analyses.

6.2.3 Design Choices

Several design choices were made for the reimplementation. The decision not to treat categorical attributes with more than 15 unique values is one of these. This threshold balances the availability of distinguishable color palettes with the cognitive limit on the number of categories a user can track visually. However, this means that attributes like "department" in an organizational network with 20 departments would be discarded from the visualization. In addition, the choice of simply assigning different color treatments to different attribute types was motivated by keeping the radial bar encoding consistent for all attributes, at the cost of making it slightly harder to differentiate between them. A more sophisticated approach might allow users to manually specify attribute types or to use alternative encodings like patterns when color alone is insufficient. A small and easy improvement would be to allow users to manually override color assignments in cases where issues occur.

For temporal networks, nodes in the circular diagram are ordered by first appearance unless dynamic reordering is applied. This creates a visual timeline effect that proved useful in the Delhi Metro use case. However, this ordering conflicts with other potentially useful orderings such as by final appearance or by an average weighting based on timestep appearances. The system allows dynamic reordering to override the chronological order, but the default chronological approach reflects a specific analytical assumption about what users care about most in temporal networks. Alternative systems might provide more explicit control over temporal ordering strategies that help users achieve their analytical requirements better.

The use of foresighted layouts in both diagram views that consider the full temporal extent of the network was implemented to help preserve mental maps during temporal navigation. However, this comes at the cost of suboptimal layouts for individual timesteps [37]. In particular, when only a small subset of nodes is active, the resulting layouts can be excessively sparse, with large portions of the available space remaining unused. This effect is

most pronounced in the circular diagram, where nodes occupy angular positions reserved for the fully developed network, leaving wide gaps between active elements. To mitigate this, the tool allows users to disable the foresighted layout for the circular diagram, enabling a more compact timestep specific arrangement that makes better use of space. In contrast, the node-link diagram always employs a foresighted layout, prioritizing positional stability over spatial efficiency.

As described in Section 4.5.2.1, change indicators reflect the previously viewed state rather than the chronologically preceding one, meaning that navigating backward from $t + 1$ to t reflects the inverse transition rather than a fixed forward direction. While an alternative design could anchor all change indicators to the chronologically preceding year regardless of navigation direction, this would create a mismatch between what the user just did and what the indicators communicate. For instance, a user stepping backward to investigate an earlier state would see additions marked as removals and vice versa, which conflicts with their navigational intent. Importantly, this design also extends naturally to filtering operations, where the same indicator mechanism communicates which nodes are added or removed as a result of applying or removing filters, rather than being limited to temporal navigation alone. By reflecting the actual transition performed in both temporal and filtering contexts, the indicators remain consistent with the user's perspective and reduce the risk of misinterpretation during exploratory analysis. The trade-off is that indicators are not stable across repeated visits to the same timestep if approached from different directions, which could cause confusion in non-linear navigation patterns.

A matrix-based representation, or a hybrid approach like NodeTrix [23], was not pursued as a primary view despite its advantages for dense networks. As discussed in Section 2.4.1.1, matrix layouts avoid visual clutter and scale better than node-link diagrams in dense graphs, but trade away topological path readability in return. This trade-off conflicts with the centrality focused analytical goals of ViNCent 2.0, where understanding how nodes connect and bridge different parts of the network is central to the analysis. Furthermore, supporting multiple centrality measures in a matrix representation would either require encoding several measures within each cell, rapidly exhausting available visual channels, or producing multiple separate matrices each encoding a single measure, which would fragment the simultaneous multi-measure comparison that the radial bar approach provides within a single unified view. That said, a matrix view could complement the existing representations as an optional coordinated panel in future work, particularly for inspecting dense subgraphs identified through community detection, where the adjacency matrix's strengths are most apparent.

The inclusion of a dedicated data table view was motivated by its presence across all related tools reviewed in Section 2.5, reflecting its role as a standard complement to graphical network presentations. The data table provides a familiar tabular interface for inspecting the full dataset, and benefits directly from the attribute metadata computed by the backend, automatically presenting each attribute with its inferred type and allowing sorting, searching and pagination across all nodes and edges. However, unlike the related tools which support direct data manipulation, the current data table is read-only. Going beyond this was not pursued within the scope of this thesis, as the primary focus remained on the graphical visualization and analytical capabilities, and is instead left as a natural direction for future work, which could bring the system closer to feature parity with established tools in this regard.

6.2.4 System Comparison

Table 6.1 provides a structured comparison of ViNCent 2.0 against the original ViNCent and the three related web-based tools reviewed in Section 2.5. The comparison highlights both the contributions of the reimplementation and its remaining limitations relative to established tools.

Table 6.1: Comparison of ViNCent 2.0 with the original ViNCent implementation and related web-based network visualization tools.

Feature / Capability	**ViNCent**	**ViNCent 2.0**	**Related tools[a]**
System capabilities			
Web-based		+	+
Export (data/image)	+		+
Supported import formats	1	5	$\geqslant$2
No. of node centrality measures	17	25	0–27
No. of community det. algorithms	0	8	1
Max. reported nodes	$\approx$10k	$\approx$5k	$\geqslant$10k
Visual representations			
Layout algorithms	2	2	5–8
Node-link diagram	+	+	+
Data table view		(+)	+
Coordinated multiple views	+	+	
Edge bundling	+	+	
Multivariate attribute support			
Arbitrary node attributes		+	+
Multi-measure visualization	+	+	(+)
Numeric filtering	+	+	(+)
Categorical filtering		+	(+)
Data distributions shown	(+)	+	(+)
Data clustering	(+)		
Temporal network support			
Temporal network support		+	
Slider-based navigation		+	
Foresighted layouts		+	
Change indicators		+	
Change preview mode		+	

Note: "+" denotes full support, "(+)" denotes partial support, and a blank cell denotes no support.
[a]Related web-based tools discussed in Section 2.5: Gephi Lite [41], Cytoscape Web [42], and Polinode [43].

Regarding the system capabilities, the migration to a web-based architecture removes the installation barrier of the original Java/Prefuse implementation, aligning ViNCent 2.0 with the accessibility of the other web-based tools. The multivariate attribute section highlights where ViNCent 2.0 departs from the original system, which was limited to centrality measures with no support for arbitrary node attributes. While related tools offer partial multi-measure support through visual channel assignment, ViNCent 2.0 encodes all measures simultaneously within the radial bar representation. This means a complete importance profile can be read at a glance without any channel assignment, a difference in kind rather than degree compared to the visual mapping approaches of the related tools. The temporal support section reflects the most distinctive contribution, as none of the related web-based tools reviewed provide this capability.

A few limitations are apparent from the comparison. The related tools offer a broader range of layout algorithms, though this reflects ViNCent 2.0's focus on its specialized multi-measure circular layout rather than general purpose layout algorithms. Additionally, the data table in ViNCent 2.0 is read-only, whereas related tools support direct data manipulation such as adding, removing or modifying nodes, edges and attributes from within the interface. The maximum network size of approximately $5,000$ nodes similarly falls short of some related tools, though this is partly a deliberate consequence of the foresighted layout strategy. Computing the force-directed layout over the complete temporal network trades

rendering efficiency for spatial consistency during temporal navigation, and future optimizations such as WebGL-based rendering discussed in Section 7.2 could narrow this gap without abandoning that strategy.

6.3 Evaluation

The formative user study and the use case scenario together provide complementary but distinct forms of evidence, each contributing different insights into the system's capabilities and limitations. The user study suggests that the system's analytical depth creates a genuine learning curve, as network analysis is not always straightforward. Participants' difficulties were not primarily interface failures but reflected a mismatch between the system's assumptions about user knowledge and participants' actual conceptual background. In particular, the circular diagram displays multiple centrality measures at once, which assumes that users already understand these metrics, something participants largely did not. This raises a question about whether the design refinements motivated by the study; explanatory tooltips and metric descriptions, address the underlying issue or simply make the system easier to approach, as users who do not understand betweenness centrality may still struggle to apply a tooltip definition during exploration.

The strong preference for the node-link diagram is also worth examining. It is unclear whether this reflects a limitation of the radial bar encoding or simply participants' greater familiarity with node-link representations. Consequently, the circular diagram received little critical feedback, as participants did not engage with it in depth. The study therefore provides limited evidence of its effectiveness for the tasks it was designed to support. Evaluation with domain experts familiar with centrality analysis would be needed to draw more meaningful conclusions.

Furthermore, no controlled empirical comparison was conducted between the radial bar encoding and alternative representations such as the glyph-based or small multiples approaches discussed in Section 2.4.1.1. Whether the radial encoding genuinely supports faster or more accurate multi-measure comparison than these alternatives therefore remains an open empirical question, and constitutes a direction for future validation.

The use case scenario addresses this gap by showing that the circular diagram and the system in general does support complex multi-measure analysis in practice, and that the temporal features supported meaningful analysis including tracking network evolution, and observing how stations gained structural importance over time. However, since the scenario was conducted by the system developer, it shows what the system is capable of rather than how effectively analysts would use it. The claim that ViNCent 2.0 effectively supports multivariate and temporal network analysis is therefore better described as demonstrated potential rather than validated effectiveness.

Overall, the findings suggest that ViNCent 2.0 succeeds as a technical platform but that its value for real users remains an open question that the current evaluation cannot fully resolve. The evidence is nonetheless sufficient to characterize it as a promising research tool that warrants further investigation with domain experts.

6.4 The Work in a Wider Context

Network analysis is increasingly important across science, from tracking disease spread to optimizing infrastructure and studying social dynamics. As networks grow in complexity and temporal depth, the demand for accessible, web-based tools that go beyond static node-link diagrams increase accordingly. ViNCent 2.0 contributes to this space by combining centrality focused analysis with multivariate attribute support and temporal navigation in a single web-based platform, lowering the barrier for researchers who need to explore evolving network data without specialized software infrastructure.

However, important limitations and ethical considerations deserve acknowledgment. The system's effectiveness depends on users understanding of network concepts like centrality measures and community structure. Without proper training, users might misinterpret visualizations or draw invalid conclusions from patterns they observe. Additionally, when applied to social networks containing personal data, visualization tools must be deployed with appropriate privacy protections and ethical oversight, as network visualization might reveal sensitive information about relationships and structural positions.

Beyond privacy concerns, a particular concern is the potential for misuse or misinterpretation of centrality-based analysis in high-stakes contexts. A node ranked highly by betweenness centrality is not necessarily influential, important or trustworthy in any real-world sense, it is simply well positioned in the graph structure. When applied to social networks, this distinction matters. Labeling a person as a "key node" or "bridge" based on structural position alone could lead to flawed decisions in organizational, legal or security contexts if the limitations of the metric are not understood. Similarly, community detection algorithms partition nodes into groups based on edge connectivity, but these groups do not necessarily correspond to meaningful social communities. Treating algorithmically detected communities as ground truth about real human relationships risks oversimplifying complex social dynamics. These risks are amplified when visualization tools make such abstractions visually compelling and easy to generate, potentially making results appear more definitive or objective than they actually are. Responsible deployment of tools like ViNCent 2.0 therefore requires not only technical documentation but also guidance on the appropriate scope and limits of interpretation.

These challenges are not unique to ViNCent 2.0 but reflect broader open questions in the information visualization community about how to design tools that are both analytically powerful and interpretable by users with varying levels of expertise. Addressing them meaningfully requires ongoing collaboration between visualization researchers and the domain communities who use these tools in practice.

Chapter 7
Conclusion

This chapter concludes the thesis by summarizing the project, answering the research questions, and discussing directions for future work.

7.1 Thesis Project Summary

The goal of this thesis was to reimplement and extend ViNCent as a modern web-based visualization system that supports interactive analysis of multivariate and temporal networks, while preserving the core centrality focused functionality of the original tool. To achieve this goal, the thesis followed an interactive design-implementation-evaluation methodology grounded in information visualization research.

The work began with a literature review and analysis of the original ViNCent system and related web-based platforms, which informed the identification of analytical tasks and system requirements. Based on these insights, a web-based system architecture was designed and the system was implemented incrementally, with continuous refinement of visual encodings, interaction techniques and architectural decisions.

Evaluation was conducted using a qualitative, formative approach appropriate for exploratory visualization systems. A formative user study provided feedback on usability and interaction design, and the findings from this evaluation directly informed subsequent refinements of the system. In addition, a final use case scenario was used to demonstrate the analytical capabilities of the completed ViNCent 2.0 system. Together, this methodological process provides the foundation for answering the research questions outlined below.

RQ1: How can ViNCent's centrality analysis capabilities be modernized as a web-based visualization system?

ViNCent's centrality analysis capabilities were modernized through an architectural redesign and technology migration while preserving core analytical workflows. The original Java/Prefuse desktop application was replaced with a client-server architecture, separating computationally expensive centrality calculations in a Python/FastAPI/NetworkX backend from the TypeScript/React/D3.js frontend focused on responsive visualization. The core visualization features were translated to web technologies: the circular diagram's radial bar encoding was reimplemented using HTML5 Canvas for performance, while the node-link diagram uses SVG for stronger interaction support. Desktop interaction features including hover, selection, histogram-based filtering and dynamic reordering were adapted to web contexts while maintaining analytical functionality. Performance optimizations including Web Worker-based edge bundling, automatic label visibility on zoom, and multiple layered HTML5 Canvases ensure interactive responsiveness. The formative evaluation and use case scenario together validate that the modernized web-based system successfully preserves ViNCent's centrality analysis capabilities while eliminating installation barriers.

RQ2: How can ViNCent be extended to support the visualization and analysis of multivariate node attributes beyond centrality?

The system successfully extends ViNCent's framework through a unified approach that treats centrality measures and arbitrary node attributes with a single analytical model. The backend's attribute metadata system automatically classifies attributes by type (categorical, numerical, boolean) and computes global ranges and unique value sets across the full temporal extent of the network. Type-based visual encodings apply appropriate color schemes, discrete palettes for categorical data, uniform colors for numerical data, and custom colors for centrality measures. The filtering interface adapts to attribute type, providing histogram-based range selection for numerical attributes and bar chart-based category selection for categorical attributes. The radial bar encoding accommodates both centrality measures and node attributes as stacked segments, enabling simultaneous comparison across multiple dimensions. This unified framework enables analysts to examine structural importance alongside domain-specific metadata seamlessly, as demonstrated in the Delhi Metro use case where transit line membership, layout type and interchange status were analyzed together with multiple centrality measures.

RQ3: How can ViNCent be extended to represent temporal networks, enabling the exploration of structural and attribute-level changes across different timesteps?

The system adopts a snapshot-based representation where each timestep represents a complete network state. However, instead of storing complete network snapshots for each timestep, only the incremental changes between successive states are stored. Each snapshot is then reconstructed by applying the accumulated changes to the current network state, allowing efficient storage and transmission while preserving the full temporal structure of the data. In the circular diagram, chronological node ordering arranges nodes by their first appearance, creating a visual timeline effect while maintaining consistent angular positions across timesteps. The node-link diagram computes its force-directed layout on the complete temporal network and selectively displays elements based on the current timestep, preventing layout instability during navigation. Visual change indicators and the change preview feature make temporal transitions explicit and manageable. The temporal slider supports both sequential stepping through time and direct jumping to specific timesteps for non-consecutive comparison. The Delhi Metro use case demonstrates that this approach successfully supports temporal analysis tasks including tracking network growth phases, identifying when specific stations gained importance, and comparing structural patterns across different time periods.

RQ4: How effective is the modernized ViNCent in visualizing and analyzing multivariate and temporal networks?

The evaluation demonstrates that ViNCent 2.0 shows strong potential for supporting exploratory analysis of multivariate and temporal networks. However, without quantitative metrics, precise effectiveness measurements remain difficult to establish. The Delhi Metro use case scenario successfully demonstrated a possible analytical workflow: the system enabled identification of key interchange stations through simultaneous comparison of multiple centrality measures and categorical attributes, revealed network evolution patterns through temporal navigation with change indicators and supported fluid transitions between analytical perspectives via coordinated views. The formative user study provided complementary evidence. Participants successfully discovered core interaction features without difficulty and demonstrated ability to explore changes in the network and examine centrality patterns, though with varying efficiency. The study proved valuable by identifying concrete usability improvements around centrality measure comprehension and filter behavior which, were addressed through design refinements. The evaluation has limitations. The use case reflects a single network type and the developers familiarity with the system. The user study involved only four participants without network analysis expertise. Without quantitative metrics, comparative evaluation or expert validation, precise effectiveness claims cannot be made. However, the combined qualitative evidence suggests the system successfully pro-

vides a functional platform for interactive exploration of multivariate and temporal networks, demonstrating clear potential for supporting realistic analytical workflows.

7.2 Future Work

This thesis establishes a foundation for multivariate and temporal visualization and analysis, but several directions would strengthen and extend the system's capabilities.

7.2.1 Expanded Evaluation

The most important next step is evaluation with domain experts conducting real analytical tasks with their own data. Studies with transportation planners, social network researchers, biologists analyzing protein interaction networks and other domain specialists would provide insights into how well the system supports real research workflows. Such studies should focus on longitudinal use over multiple sessions rather than single exploratory sessions, revealing whether the system supports hypothesis formation, iterative refinements and insight generation in realistic contexts. Systematic comparison against established tools would clarify ViNCent 2.0's relative strengths and weaknesses. Quantitative task-based studies measuring completion time and accuracy for specific analytical questions could identify where the circular diagram's multi-measure encoding provides advantages over traditional approaches and where it introduces overhead. In addition, evaluation with diverse network types such as social, biological and citation networks, of varying sizes and temporal characteristics would establish generalizability and reveal scalability limits.

7.2.2 Analytical Features

Reimplementing clustering based on centrality profiles would restore a feature from the original ViNCent that is missing in ViNCent 2.0. This would help identify groups of structurally similar nodes in the network, complementing the existing community detection which groups nodes based on edge connectivity rather than attribute similarity. Possible extensions could enable clustering to operate across all attribute types, allowing analysts to group nodes by arbitrary combinations of centrality measures and domain-specific metadata simultaneously. For instance, clustering by both betweenness centrality and a categorical attribute such as node type would reveal whether certain structural roles tend to align with domain categories, further supporting the kind of combined structure-attribute analysis that motivated the multivariate extensions in this thesis. Exposing cluster membership as a node attribute would further allow users to filter and reorder the circular diagram by cluster assignment, integrating clustering naturally into the existing analytical workflow. While community visualization was added after the user study, deeper integration would be valuable. Community-level metrics that aggregate node properties within communities, such as mean centrality or dominant categorical attributes per community, would further support multi-level analysis by allowing analysts to characterize and compare groups rather than only individual nodes.

Beyond cluster and community visualization, the node-link diagram currently supports size and color mappings for arbitrary attributes, but does not provide dedicated visual encodings for these measures comparable to the radial bar representation in the circular diagram. Future work could introduce attribute-specific encodings directly within the node-link view, such as rendering concentric rings round nodes to encode multiple attribute dimensions simultaneously, or using pie-chart style node glyphs similar to those explored in prior multivariate network visualization work. This would reduce the need for users to switch between views when interpreting structural importance in a topological context, addressing the key finding from the user study where participants expressed a strong preference working with the node-link diagram. A particular challenge would be balancing visual expressiveness with

readability, as dense networks risk occlusion when nodes carry complex encodings. Adaptive encodings that simplify at lower zoom levels and reveal full centrality detail on closer inspection could mitigate this. Integrating such encodings would bring the node-link diagram closer to feature parity with the circular diagram for centrality focused tasks, while retaining its advantage in conveying topological structure.

The temporal dimension of the system also offers room for improvement as the current visualization focuses on tracking presence, absence and value changes across snapshots, but does not provide dedicated encodings for how a node's relative ranking evolves over time. Future work could introduce rank-based encodings within the circular diagram by incorporating trajectory encodings such as small embedded sparklines showing rank history directly on each radial bar segment, thereby supporting longitudinal analysis without requiring users to step manually through timesteps. Beyond visual encodings, computing derived temporal metrics such as rate of centrality change, stability scores indicating how consistently nodes maintain their rankings, and appearance/disappearance frequency would enrich temporal analysis. Adding specialized visual encodings to better track community evolution, such as when communities split, merge or persist across timesteps, alongside a community stability metric quantifying how consistently a set of nodes remains grouped together over time, would provide an additional dimension for understanding structural evolution. The system could additionally encode summary attributes such as rank volatility, the variance of a node's ranking across all timesteps or whether a node consistently ranks among the top performers. Implementing temporal queries like "find nodes whose betweenness increased by more than 50% between timesteps 5 and 10" would further support targeted exploration. Extending the snapshot model to support temporal aggregations at user-specified scales such as hourly, daily or monthly groupings would allow analysts to zoom out to high-level trends before drilling into finer grained periods of interest, supporting exploration across multiple temporal scales within the same system. These extensions would more directly strengthen the support for tasks T3 and T4 identified in Section 3.1.1.2, where analysts must compare network states across timesteps, identify trends in network evolution and formulate explanations for observed structural changes.

7.2.3 Technical Improvements

Two important improvements concern how the system classifies and presents attributes. First, the backend currently determines attribute types automatically based on the number of unique values and value formats. While this works well in most cases, it can misclassify attributes, for instance failing to recognize a sparsely populated categorical field. Allowing users to manually override the inferred attribute type through the data loading interface or data table view would give analysts direct control over how attributes are encoded and filtered, reducing confusion arising from incorrect automatic classification. Second, the system currently restricts categorical visualization to attributes with at most 15 unique values. Future work could allow users to manually select which specific values within a large categorical attribute they wish to visualize, effectively treating the remainder as an "other" category. This would make the system applicable to datasets with a larger number of categorical attributes such as organizational departments, geographic regions or product categories, which are common in real-world network data but currently fall outside the system's supported range.

Related to data loading, the system processes the complete temporal network upfront before analysis begins, computing all centrality measures and community memberships across every timestep at load time. While this works well for smaller datasets, networks with hundreds of timesteps or large node sets can result in long load times and significant frontend memory consumption. Future work could introduce a streaming or on-demand fetching model, where centrality computations and network snapshots are fetched incrementally as the user navigates through timesteps rather than precomputed in full. This would substan-

tially reduce both initial load time and memory overhead, making the system practical for much larger temporal datasets. Caching previously fetched timesteps locally would further improve navigation responsiveness by avoiding redundant recomputation when revising earlier states. However, this approach would require rethinking several architectural assumptions. Both the foresighted layout strategy and the attribute metadata system currently depend on knowing the full temporal extent of the network in advance. The layout strategy would need to be adapted to work with partially loaded data, for instance by computing layouts based on a representative sample or the currently loaded portion of the network, while the attribute system would need to support incremental classification that can be revised or refined as additional timesteps are fetched. These adaptations introduce a fundamental trade-off: layouts and color encodings computed on partial data may shift or be reclassified as more timesteps are loaded, potentially introducing visual inconsistencies that disrupt the spatial stability and encoding coherence the system currently guarantees.

Beyond data loading, current system performance limits practical use to networks with just a few thousand nodes and edges. Implementing level-of-detail rendering, where distant or small nodes in both diagram views are simplified or aggregated based on zoom level, would improve responsiveness for larger networks by reducing the number of elements rendered at any given time. The circular diagram in particular renders all radial bar segments regardless of their current visibility or size on screen, meaning that large networks incur full rendering costs even when most bars and segments are too small to inspect meaningfully. GPU-accelerated layout computation and rendering using WebGL could enable interactive layouts for networks with potentially above tens of thousands of elements, replacing the current SVG and Canvas-based rendering which is inherently limited by the CPU. The force-directed layout in the node-link diagram is particularly computationally expensive for large networks, and offloading simulation calculations to the GPU or replacing them with approximate methods would significantly improve performance. Deploying the system for multi-user access requires addressing several architectural limitations. Request timeout mechanisms and workload queuing would prevent single large computations from blocking the backend and affecting other users. Session-based result caching would allow users to resume analyses without recomputation, while database integration for persistent storage would enable saving analyses and sharing results with collaborators.

Bibliography

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

[2] A. Kerren, H. Köstinger, and B. Zimmer. "ViNCent – Visualization of Network Centralities". In: *Proceedings of the International Conference on Information Visualization Theory and Applications (IVAPP '12)*. INSTICC, 2012, pp. 703–712. DOI: 10.5220/0003822207030712.

[3] B. Zimmer, I. Jusufi, and A. Kerren. "Analyzing Multiple Network Centralities with ViNCent". In: *Proceedings of SIGRAD 2012: Interactive Visual Analysis of Data*. Växjö, Sweden, 2012, pp. 87–90. URL: https://ep.liu.se/en/conference-article.aspx?series=ecp%5C&issue=81%5C&Article_No=12.

[4] M. Sedlmair, M. Meyer, and T. Munzner. "Design Study Methodology: Reflections from the Trenches and the Stacks". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2431–2440. DOI: 10.1109/TVCG.2012.213.

[5] T. Munzner. *Visualization Analysis and Design*. AK Peters Visualization Series. Boca Raton, FL: CRC Press, 2014.

[6] L. C. Freeman. "Centrality in social networks conceptual clarification". In: *Social Networks* 1.3 (1978), pp. 215–239. DOI: 10.1016/0378-8733(78)90021-7.

[7] S. Fortunato. "Community detection in graphs". In: *Physics Reports* 486.3 (2010), pp. 75–174. DOI: https://doi.org/10.1016/j.physrep.2009.11.002.

[8] A. Lancichinetti and S. Fortunato. "Community detection algorithms: A comparative analysis". In: *Phys. Rev. E* 80 (5 Nov. 2009), p. 056117. DOI: 10.1103/PhysRevE.80.056117.

[9] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. "Fast Unfolding of Communities in Large Networks". In: *Journal of Statistical Mechanics Theory and Experiment* 2008 (Apr. 2008). DOI: 10.1088/1742-5468/2008/10/P10008.

[10] U. N. Raghavan, R. Albert, and S. Kumara. "Near linear time algorithm to detect community structures in large-scale networks". In: *Phys. Rev. E* 76 (3 Sept. 2007), p. 036106. DOI: 10.1103/PhysRevE.76.036106.

[11] U. von Luxburg. "A Tutorial on Spectral Clustering". In: *CoRR* abs/0711.0189 (2007). URL: http://arxiv.org/abs/0711.0189.

[12] M. E. J. Newman and M. Girvan. "Finding and evaluating community structure in networks". In: *Phys. Rev. E* 69 (2 Feb. 2004), p. 026113. DOI: 10.1103/PhysRevE.69.026113.

[13] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. "Task taxonomy for graph visualization". In: *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. BELIV '06. Venice, Italy: Association for Computing Machinery, 2006, pp. 1–5. DOI: 10.1145/1168149.1168168.

[14] S. P. Borgatti. "Centrality and network flow". In: *Social Networks* 27.1 (2005), pp. 55–71. DOI: 10.1016/j.socnet.2004.11.008.

[15] M. J. Newman. “A measure of betweenness centrality based on random walks”. In: *Social Networks* 27.1 (2005), pp. 39–54. DOI: 10.1016/j.socnet.2004.11.009.

[16] J. Pretorius, H. C. Purchase, and J. T. Stasko. “Tasks for Multivariate Network Analysis”. In: *Multivariate Network Visualization: Dagstuhl Seminar #13201, Dagstuhl Castle, Germany, May 12-17, 2013, Revised Discussions*. Ed. by A. Kerren, H. C. Purchase, and M. O. Ward. Cham: Springer International Publishing, 2014, pp. 77–95. DOI: 10.1007/978-3-319-06793-3_5.

[17] B. Shneiderman. “The eyes have it: A task by data type taxonomy for information visualizations”. In: *Proceedings 1996 IEEE Symposium on Visual Languages*. 1996, pp. 336–343. DOI: 10.1109/VL.1996.545307.

[18] J. C. Roberts. “State of the Art: Coordinated Multiple Views in Exploratory Visualization”. In: *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*. 2007, pp. 61–71. DOI: 10.1109/CMV.2007.20.

[19] U. Brandes, M. Eiglsperger, M. Kaufmann, J. Lerner, and C. Pich. *GraphML Specification*. http://graphml.graphdrawing.org. Accessed: 2026-01-26. 2013.

[20] A. Kerren, H. C. Purchase, and M. O. Ward. “Introduction to Multivariate Network Visualization”. In: *Multivariate Network Visualization: Dagstuhl Seminar #13201, Dagstuhl Castle, Germany, May 12-17, 2013, Revised Discussions*. Ed. by A. Kerren, H. C. Purchase, and M. O. Ward. Cham: Springer International Publishing, 2014, pp. 1–9. DOI: 10.1007/978-3-319-06793-3_1.

[21] J. Pearlman and P. Rheingans. “Visualizing Network Security Events Using Compound Glyphs from a Service-Oriented Perspective”. In: *VizSEC 2007. Mathematics and Visualization*. 2007, pp. 131–146. DOI: 10.1007/978-3-540-78243-8_9.

[22] M. Ghoniem, J.-D. Fekete, and P. Castagliola. “On the Readability of Graphs Using Node-Link and Matrix-Based Representations: A Controlled Experiment and Statistical Analysis”. In: *Information Visualization Journal* 4 (May 2005). DOI: 10.1057/palgrave.ivs.9500092.

[23] N. Henry, J.-D. Fekete, and M. J. McGuffin. “NodeTrix: a Hybrid Visualization of Social Networks”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1302–1309. DOI: 10.1109/TVCG.2007.70582.

[24] I. Safarli and A. Lex. “TaMax: Visualizing Dense Multivariate Networks with Adjacency Matrices”. In: *IEEE VIS Posters*. Poster. 2019. URL: https://sci.utah.edu/~vdl/papers/2019_infovis_tamax.pdf.

[25] D. Archambault, H. Purchase, and B. Pinaud. “Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.4 (2011), pp. 539–552. DOI: 10.1109/TVCG.2010.78.

[26] A. Bezerianos, F. Chevalier, P. Dragicevic, N. Elmqvist, and J. Fekete. “GraphDice: A System for Exploring Multivariate Social Networks”. In: *Computer Graphics Forum* 29.3 (2010), pp. 863–872. DOI: 10.1111/j.1467-8659.2009.01687.x.

[27] P. Holme and J. Saramäki. “Temporal networks”. In: *Physics Reports* 519.3 (2012). Temporal Networks, pp. 97–125. DOI: 10.1016/j.physrep.2012.03.001.

[28] P. Holme. “Modern temporal network theory: a colloquium”. In: *The European Physical Journal B* 88.9 (2015), pp. 1–30. DOI: 10.1140/epjb/e2015-60657-4.

[29] B. Bach, E. Pietriga, and J.-D. Fekete. “Visualizing dynamic networks with matrix cubes”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, pp. 877–886. DOI: 10.1145/2556288.2557010.

[30] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. "Reordering Massive Sequence Views: Enabling temporal and structural analysis of dynamic networks". In: *2013 IEEE Pacific Visualization Symposium (PacificVis)*. 2013, pp. 33–40. DOI: 10.1109/PacificVis.2013.6596125.

[31] D. Keim, J. Schneidewind, and M. Sips. "CircleView: A New Approach for Visualizing Time-related Multidimensional Data Sets". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '04. New York, NY, USA: Association for Computing Machinery, 2004, pp. 179–182. DOI: 10.1145/989863.989891.

[32] C. D. G. Linhares, J. R. Ponciano, J. G. S. Paiva, L. E. C. Rocha, and B. A. N. Travençolo. "DyNetVis - An interactive software to visualize structure and epidemics on temporal networks". In: *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2020, pp. 933–936. DOI: 10.1109/ASONAM49781.2020.9381304.

[33] C. D. G. Linhares, J. R. Ponciano, D. S. Pedro, L. E. C. Rocha, A. J. M. Traina, and J. Poco. "LargeNetVis: Visual Exploration of Large Temporal Networks Based on Community Taxonomies". In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (2023), pp. 203–213. DOI: 10.1109/TVCG.2022.3209477.

[34] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. "A Taxonomy and Survey of Dynamic Graph Visualization". In: *Computer Graphics Forum* 36.1 (2017), pp. 133–159. DOI: 10.1111/cgf.12791.

[35] J. Moody and D. Mcfarland. "Dynamic Network Visualization1". In: *American Journal of Sociology - AMER J SOCIOL* 110 (Jan. 2005), pp. 1206–1241. DOI: 10.1086/421509.

[36] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. "A Taxonomy and Survey of Dynamic Graph Visualization". In: *Computer Graphics Forum* 36.1 (2017), pp. 133–159. DOI: 10.1111/cgf.12791.

[37] S. Diehl, C. Görg, and A. Kerren. "Preserving the Mental Map using Foresighted Layout". In: *Proceedings of Joint Eurographics - IEEE TVCG Symposium on Visualization*. VisSym '01. 2001. DOI: 10.2312/VisSym/VisSym01/175-184.

[38] A. Lhuillier, C. Hurter, and A. Telea. "State of the Art in Edge and Trail Bundling Techniques". In: *Computer Graphics Forum* 36.3 (2017), pp. 619–645. DOI: 10.1111/cgf.13213.

[39] M. Rosvall and C. T. Bergstrom. "Mapping Change in Large Networks". In: *PLOS ONE* 5.1 (Jan. 2010), pp. 1–7. DOI: 10.1371/journal.pone.0008694.

[40] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. "Reordering Massive Sequence Views: Enabling temporal and structural analysis of dynamic networks". In: *2013 IEEE Pacific Visualization Symposium (PacificVis)*. 2013, pp. 33–40. DOI: 10.1109/PacificVis.2013.6596125.

[41] M. Bastian, S. Heymann, and M. Jacomy. "Gephi: An Open Source Software for Exploring and Manipulating Networks". In: *Third International ICWSM Conference*. 2009, pp. 361–362.

[42] K. Ono, D. Fong, C. Gao, C. Churas, R. Pillich, J. Lenkiewicz, D. Pratt, A. R. Pico, K. Hanspers, Y. Xin, J. Morris, M. Kucera, M. Franz, C. Lopes, G. Bader, T. Ideker, and J. Chen. "Cytoscape Web: bringing network biology to the browser". In: *Nucleic Acids Research* 53.W1 (May 2025), W203–W212. DOI: 10.1093/nar/gkaf365.

[43] A. Pitts. "Polinode: A web application for the collection and analysis of network data". In: *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2016, pp. 1422–1425. DOI: 10.1109/ASONAM.2016.7752435.

[44] T. Munzner. "A Nested Model for Visualization Design and Validation". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 921–928. DOI: 10.1109/TVCG.2009.111.

[45] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. "Empirical Studies in Information Visualization: Seven Scenarios". In: *IEEE Transactions on Visualization and Computer Graphics* 18.9 (Sept. 2012), pp. 1520–1536. DOI: 10.1109/TVCG.2011.279.

[46] J. R. Lewis. "Usability Testing". In: *Handbook of Human Factors and Ergonomics*. John Wiley Sons, Ltd, 2006. Chap. 49, pp. 1275–1316. DOI: https://doi.org/10.1002/0470048204.ch49.

[47] R. A. Virzi. "Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough?" In: *Human Factors* 34.4 (1992), pp. 457–468. DOI: 10.1177/001872089203400407.

[48] Meta Platforms, Inc. *React Documentation*. Accessed: 2026-01-26. 2026. URL: https://react.dev/reference/react.

[49] Microsoft. *TypeScript Handbook*. Accessed: 2026-01-26. 2026. URL: https://www.typescriptlang.org/docs/handbook/intro.html.

[50] M. Bostock, V. Ogievetsky, and J. Heer. "D³ Data-Driven Documents". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (Dec. 2011), pp. 2301–2309. DOI: 10.1109/TVCG.2011.185.

[51] M. Bostock. *D3.js Documentation*. Accessed: 2026-01-26. 2026. URL: https://d3js.org/.

[52] pmndrs. *Zustand Documentation*. Accessed: 2026-01-26. 2026. URL: https://docs.pmnd.rs/zustand/getting-started/introduction.

[53] Redux Team. *Redux FAQ: When Should I Use Redux?* Accessed: 2026-01-26. 2026. URL: https://redux.js.org/faq/general#when-should-i-use-redux.

[54] Shadcn. *shadcn/ui Documentation*. Accessed: 2026-01-25. 2026. URL: https://ui.shadcn.com/docs.

[55] Tailwind Labs. *Tailwind CSS Documentation*. Accessed: 2026-01-25. 2026. URL: https://tailwindcss.com/docs.

[56] MUI Team. *Material UI Documentation*. Accessed: 2026-01-26. 2026. URL: https://mui.com/material-ui/getting-started/overview/.

[57] S. Ramis. *FastAPI Documentation*. Accessed: 2026-01-26. 2026. URL: https://fastapi.tiangolo.com/.

[58] A. A. Hagberg, D. A. Schult, and P. J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by G. Varoquaux, T. Vaught, and J. Millman. Pasadena, CA USA, 2008, pp. 11–15.

[59] NetworkX Developers. *NetworkX Documentation*. Accessed: 2026-01-26. 2026. URL: https://networkx.org/documentation/stable/.

[60] NetworkX Developers. *NetworkX Centrality Algorithms — Official Documentation*. Accessed: 2026-01-26. 2026. URL: https://networkx.org/documentation/stable/reference/algorithms/centrality.html.

[61] NetworkX Developers. *NetworkX Community Detection Algorithms — Official Documentation*. Accessed: 2026-01-26. 2026. URL: https://networkx.org/documentation/stable/reference/algorithms/community.html.

[62] R. Bhatia. *Star Wars Social Network*. https://www.kaggle.com/datasets/ruchi798/star-wars. Accessed: 2026-01-28. Kaggle, 2026.

[63] A. Jangir. *Delhi Metro Dataset*. https://www.kaggle.com/datasets/arunjangir245/delhi-metro-dataset. Accessed: 2026-01-27. Kaggle, 2026.

[64] M. B. Rosson and J. M. Carroll. "Scenario-Based Design". In: *The Human–Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. Ed. by J. A. Jacko and A. Sears. Originally published in 2002. Mahwah, NJ, USA: Lawrence Erlbaum Associates, 2002, pp. 1032–1050.

[65] N. I. of Standards and Technology. *Matrix Market: File Formats*. https://math.nist.gov/MatrixMarket/formats.html. Accessed: 2026-01-26. 2013.

[66] C. A. Brewer. *ColorBrewer: Color Advice for Maps*. https://colorbrewer2.org. Accessed: 2026-01-26. 2025.

[67] P. Green-Armytage. "A Colour Alphabet and the Limits of Colour Coding". In: *Color: Design Creativity* 5 (Aug. 2010), pp. 1–23.

[68] D. Holten and J. J. Van Wijk. "Force-Directed Edge Bundling for Graph Visualization". In: *Computer Graphics Forum* 28.3 (2009), pp. 983–990. DOI: 10.1111/j.1467-8659.2009.01450.x.

[69] D. Archambault and H. Purchase. "Mental Map Preservation Helps User Orientation in Dynamic Graphs". In: *Graph Drawing. GD 2012*. 2012. DOI: 10.1007/978-3-642-36763-2_42.

[70] D. Archambault, H. Purchase, and B. Pinaud. "Difference Map Readability for Dynamic Graphs". In: *Graph Drawing. GD 2010*. 2010. DOI: 10.1007/978-3-642-18469-7_5.

[71] J. Nielsen. "Iterative user-interface design". In: *Computer* 26.11 (1993), pp. 32–41. DOI: 10.1109/2.241424.

[72] B. Shneiderman and C. Plaisant. "Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies". In: *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. BELIV '06. Venice, Italy: Association for Computing Machinery, 2006, pp. 1–7. DOI: 10.1145/1168149.1168158.

[73] S. Carpendale. "Evaluating Information Visualizations". In: *Information Visualization: Human-Centered Issues and Perspectives*. Ed. by A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 19–45. DOI: 10.1007/978-3-540-70956-5_2.

Appendix A
ViNCent 2.0 User Study: Interview Template

Estimated Duration: 40–50 minutes

A.1 Introduction (2–3 minutes)

- Present myself and thank the participant
- Explain purpose: evaluating ViNCent 2.0 for master's thesis
- Outline session structure
- Emphasize thinking aloud during the session

A.2 Background Information (5 minutes)

Ask the participant:

- About their background
- About their experience with network visualization
- Their experience level with networks
- Their knowledge of centrality measures
- Their familiarity with Star Wars

A.3 Basic Interaction Discovery (5 minutes)

Send the page link and ask the participant to share their screen and load the Star Wars network.

Ask them to try the following:

- Zooming and panning the visualization
- Switching between different views (circular, node-link, data table)
- Hovering over nodes and edges to see what happens
- Selecting and multi-selecting nodes in both the circular and node-link views
- Finding tooltip information about characters
- Navigating between different time periods (episodes)

Observe how intuitively they discover the basic interactions.

Follow up Questions

- Which interactions felt natural?
- Was anything hard to figure out?

A.4 Discovering and Understanding Features (15–20 minutes)

Ask the participant to think aloud while exploring the system.
Ask the following questions:

Task 1: Can you identify which characters appear in multiple episodes versus only one episode?

Task 2: Find all characters that are highly connected (have many relationships) but only appear in one or two episodes.

Task 3: Can you identify which characters act as bridges between different groups in the network?

Task 4: Compare the network structure between Episode IV and Episode VI. What are the main differences?

Task 5: Find characters who increase in importance (centrality) across the episodes.

Task 6: Reorder the nodes in the circular view to group characters by their most dominant centrality measure. What patterns do you notice?

For each question, observe which features they use.

Follow up Questions

- Which features were most helpful?
- Which tasks felt difficult and why?
- Was there any information they wanted but could not find?
- Were any features discovered by accident?

A.5 General Usability Discussion (5–10 minutes)

Ask about their thoughts on the system and overall experience, including:

- Overall experience and first impressions
- Most helpful features
- Most frustrating or challenging aspects
- Unclear or confusing elements
- Preference between circular diagram and node-link diagram
- Clarity and usefulness of temporal navigation
- Feeling of control over the visualization
- Color usage and clarity
- Overall usability rating (1–10) and reasoning
- Any other suggestions for improvement

A.6 Closing (5 minutes)

- Answer any remaining questions
- Ask for final suggestions or comments
- Thank the participant