VISUALIZATION OF DYNAMIC GRAPHS

YUNZHE WANG

PhD

The Hong Kong Polytechnic University

2020

The Hong Kong Polytechnic University

Department of Computing

Visualization of Dynamic Graphs

Yunzhe Wang

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

August 2019

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

(Signed)

Yunzhe Wang (Name of student)

ABSTRACT

A dynamic graph is built from a Cartesian set $V(t) \times E(t)$ where $V(t)$ and $E(t)$ denote the set of vertices and edges at time t, respectively. Since changes often happen at massive local parts of the graph, it is difficult to capture and understand them. Visualization of dynamic graphs can alleviate the difficulty as it maps changes to graphics that can be better perceived by people.

In visualization, graphs are usually drawn as node-link or matrix diagrams, and the temporal dimension is represented by timelines or animations. Based on these visualization techniques, various developments have been made, from providing a layout algorithm that optimizes the visual stability to applying analysis to real-world datasets of different disciplines. This thesis aims at investigating previous work in the area of visualizing dynamic graphs, and then concentrating on discovering structural and semantical patterns from dynamic graphs. Specific contributions are as follows.

First, we provide a method of searching large graphs for special topologies. The method conducts Community Detection to obtain components of manageable sizes, then classifies them according to their structures. *Second*, we investigate the dynamics of attributes of graph entities. Specifically, we implement an application of deriving functionalities of geographical regions by analysing a temporal network constructed from geo-textual data. Natural Language Processing techniques are used to deal with the textual part of attributes. *Third*, we improve the usability of traditional animations and timelines. To help users compare adjacent frames of the animation, glyphs of two timestamps are placed concentrically in one view. Meanwhile, visual changes caused by the glyph transformation is minimized by a pentagonal design. In timelines, identical objects at all timestamps are identified and linked chronologically to facilitate the tracing of object evolution.

PUBLICATIONS ARISING FROM THE THESIS

1. **Yunzhe, Wang**, George Baciu, and Chenhui Li. Visualizing Functional Regions by Analysis of Geo-textual Data. In Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers, pages 25–29. Eurographics Association, 2018.

2. **Yunzhe, Wang**, George Baciu, and Chenhui Li. Cognitive Visualization of Popular Regions Discovered From Geo-Tagged Social Media Data. International Journal of Cognitive Informatics and Natural Intelligence (IJCINI) 12.1 (2018): 14-28.

3. **Yunzhe, Wang**, George Baciu, and Chenhui Li. Smooth Animation of Structure Evolution in Time-Varying Graphs with Pattern Matching (**Best Paper Award**). SIGGRAPH Asia 2017 Symposium on Visualization. ACM, 2017.

4. **Yunzhe Wang**, George Baciu, and Chenhui Li. Cognitive Exploration of Regions through Analyzing Geo-Tagged Social Media Data. In16th IEEE International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC 2017, Oxford,United Kingdom, July 26-28, 2017, pages 59–64, 2017

5. **Yunzhe Wang**, George Baciu, and Chenhui Li. Visualizing the Temporal Similarity Between Clusters of Dynamic Graphs. In18th IEEE International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC 2019. (Accepted)

6. Chenhui Li, George Baciu, and **Yunzhe Wang**. ModulGraph: Modularity-Based Visualization of Massive Graphs. SIGGRAPH Asia 2015 Symposium On Visualization in High Performance Computing, 2015.

7. George Baciu, Chenhui Li, **Yunzhe Wang**, and Xiujun Zhang. Cloudets: Cloud-Based Cognition for Large Streaming Data (**Best Paper Award**). 14th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI* CC2015), 2015.

8. George Baciu, **Yunzhe Wang**, and Chenhui Li. Cognitive Visual Analytics of Multi-Dimensional Cloud System Monitoring Data. 15th IEEE International Conference on Cognitive Informatics and Cognitive Computing, 2016.

9. Chenhui Li, George Baciu, and **Yunzhe Wang**. Visquery: Visual Querying of Streaming Data via Pattern Matching. InDigital Media Industry & Academic Forum, DMIAF 2016, Santorini, Greece, July 4-6, 2016, pages 161–165, 2016.

10. Chenhui Li, George Baciu, and **Yunzhe Wang**. Module-based Visualization of Large-scale Network. Journal of Visualization. 2016.

11. George Baciu, Chenhui Li, **Yunzhe Wang**, and Xiujun Zhang. A Cloud-Driven Visual Cognition of Large Streaming Data. International Journal of Cognitive Informatics and Natural Intelligence, 2016, 10(1): 12-31.

12. Chenhui Li, George Baciu, **Yunzhe Wang**, and Xiujun Zhang. Fast Content-Aware Resizing of Multi-Layer Information Visualization via Adaptive Triangulation. Journal of Visual Languages and Computing, 2017.

ACKNOWLEDGEMENTS

I would like to express the highest respect and gratefulness to my supervisor, Prof. George Baciu, who gave me the opportunity to start the Ph.D. research. He teaches me a lot on how to think rather than just tell me what to do. The rigorous attitude and the research skills that I learn from him will benefit me for the rest of my life. Besides, I am deeply appreciative of his patience and responsibility for helping me to revise this thesis.

I also want to thank all the previous GAMA Lab memebers for their help. Especially, Dr. Chenhui Li, his selflessness and kindness are examples for me to learn. Without his inspiration and help, I would have been stuck in detours and could not find a way out.

Last but not least, I want to show my greatest love to my parents. They are always there, encouraging me when I am frustrated and giving me suggestions when I am confused. Their love and accompany relieve me from the stress and support me to complete this research study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Data visualization is a broad field that involves two main streams: information visualization and scientific visualization [1]. Scientific visualization is primarily about rendering scientific phenomena. It emphasizes factors like volumes and surfaces. This thesis focuses on information visualization which aims at representing abstract data by specially designed visual encodings. Through visual cognition, users can easily obtain an understanding of data, and they can further perform analysis with the help of interactive tools.

The target data of this thesis are dynamic graphs, which are also known as time-varying graphs. We note that we interchangeably use one of the following pairs of terms: (*graph*, *network*), (*node*, *vertex*) and (*link*, *edge*). A graph is an important structure that models the relationship between objects. Data collected from different sources like software systems and social media applications can be abstracted as dynamic graphs. For example, we can map Facebook users and their real-time interactions to graph vertices and edges, respectively.

Dynamic graphs can be depicted by the *snapshot* or *temporal* models [2]. For the temporal model, no aggregation of time is performed, and nodes and links are labeled by timestamps indicating when they appear and disappear. For the snapshot model, a

snapshot either stands for the graph state at a specific time or the aggregation of multiple states during a period. Hence, there is an information loss because changes that happen within the period are concealed. The time granularity between snapshots is often decided by contextual needs and it affects the result of analysis. For example, by categorizing events by hour, day or week, security management officers can inspect their occurrence at different frequencies. This thesis adopts the snapshot model, because it reduces the complexity of analysis and allows for parallel processing. A dynamic graph G in the snapshot model can be defined as an ordered set: $\{G_1, G_2, ..., G_n\}$, where each snapshot $G_i = (V_i, E_i)$ is identified by the sets of nodes V_i and edges E_i, and $E_i \subseteq V_i \times V_i$.

Graph dynamics represent the dynamic phenomena that occur on graphs. For simple graphs [3], the insertion and deletion of vertices and edges makes the graph topology vary over time as shown in the example in Figure 1.1. Small circles in the node-link diagrams denote vertices and the straight lines represent connections between vertices. For real-world networks, dynamic phenomena may also include the variation of the attributes of vertices and edges. For example, the interaction frequency between social media application users changes at different time. Our goal is to reveal the evolutionary patterns of both graph entities and their attributes by graphical representations.

1.1 Objectives

In the context of visual analytics of complex graphs, there exist various challenges, and we are interested in providing solutions to two of them. First, how to satisfy the specific

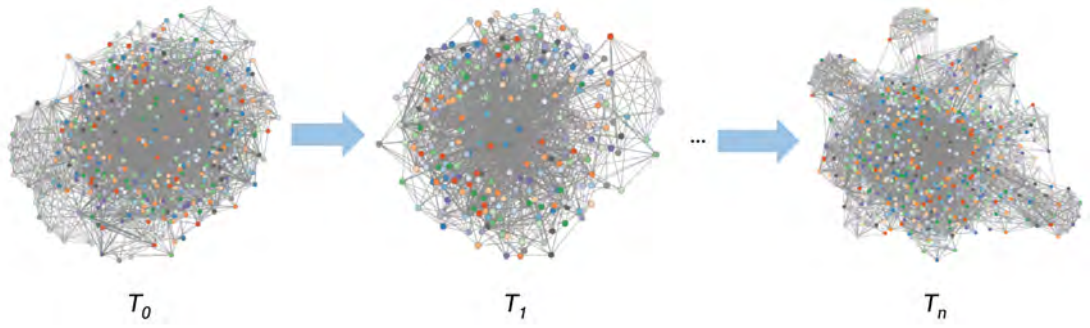


Figure 1.1: Snapshots of a dynamic graph at three time steps.

goal of analysis in different scenarios? For example, security engineers wish to find the vulnerabilities of the computer systems, while biologists might be interested in detecting the anomalies from cell communications. Second, how to interpret the result of analysis? Especially when users lack expertise.

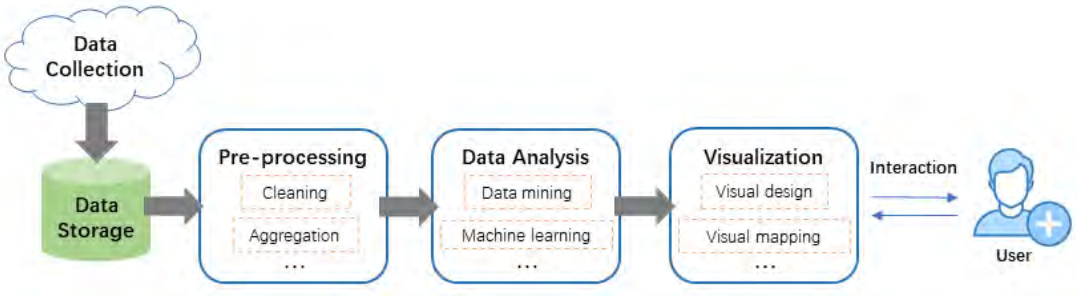


Figure 1.2: The pipeline of building a visual system.

Our solution is to build various visual systems that fulfill the requirements of specific exploration tasks. As shown in Figure 1.2, the building procedure involves a few essential stages. These systems incorporate powerful techniques from data mining, machine learning and other disciplines. They also represent data by well-designed visual encodings that are easier to perceive than abstract numbers or descriptions. The specific objectives of this

thesis are as follows:

1. *To expose the structural similarities between graph components*. We divide graphs by grouping closely related entities into components. Then, we can focus on local patterns that are easier to analyse due to the smaller scale. However, it is also important to discover how those components interact with each other. We want to inspect whether some of them possess similar structures and expose the evolutionary patterns of the similarity;

2. *To search for the topologies of interest from large graphs*. Based on the same idea of dividing graphs into components, we aim at making the topologies of these components searchable so that users can quickly locate the patterns they are interested in. Meanwhile, it will be convenient to track the structural evolution of components;

3. *To represent the dynamics of attributed graphs*. Entities have intrinsic characteristics in attributed graphs. For example, in peoples' mobility graphs, vertices that denote geographical venues have longitude-latitude coordinates. To learn the evolution of graphs, we can analyse different types of characteristics and their changes by effective tools;

4. *To improve the usability of traditional representations for dynamic graphs*. Due to the limited space of the display, static node-link diagrams suffer from overlapping issues when there are too many vertices and edges. The temporal dimension of dynamic graphs can be mapped to timelines or animations that both have pros and

cons. For animations, users can hardly remember continuous frames, not to mention comparing them. For timelines, they are often criticized for the poor scalability. Our mission is to integrate traditional methods and exploit their strengths. Meanwhile, we need to provide a rich set of interactive features to facilitate the data exploration.

1.2 Contributions

We established a cloud platform to promote the efficient data processing [4, 5, 6]. We classified graph components based on their structures so that users can search for topologies of interest [7, 8]. The traditional animated representations were improved to facilitate visual comparisons [9]. By studying the semantics of dynamic networks that are constructed from temporal geo-textual data, we were able to track the evolution of geographical regions [10, 11, 12]. Detailed contributions are as follows:

- *A framework for exposing the structual similarities between graphs.* The framework consists of three main modules, which are *community detection*, *graph vectorization* and *visualization*. Each module offers multiple advanced methods for implementation. We map the similarities between communities to the distances between nodes in graph drawings so as to let users obtain a visual understanding about which communities have similar structures;

- *Classification of complex graph structures.* We reduce the diversity of graph structures by classifying them to topological patterns. Two methods are available for

classification. The first one relies on computing a set of topological properties. We represent graphs by the aggregation of the properties, and then compute the structural similarities based on these new representations. The second method is based on the fact that a determinate layout algorithm generates similar appearances for graphs having similar structures. We take layout images rather than original graphs as the input and train a classification model. Consequently, the topological pattern of an arbitrary graph can be predicted without exhaustive searches and calculations;

- *Learning semantics from dynamic geo-textual graphs.* We apply the frequently used techniques in graph visualizations to solving problems in the geographical domain. Firstly, graphs are constructed from discrete data items. These graphs encode both the spatial and semantic information of venues. We then conduct community detection algorithms on graphs to delimit functional regions. To present the evolution of regions, we integrate two complementray views, one for displaying animations and the other for showing juxtaposed snapshots. Study results demonstrate that our method successfully helps users to capture a few types of regional events, such as appear, disappear, expand, shrink, etc;

- *Smooth and simplified visualizations of large dynamic graphs.* For the purpose of visual simplicity, our graph visualizations are based on the collapsed node-link diagrams where a single node represents a cluster of entities. However, such diagrams hide the local connectivity information from view. Therefore, we enrich the visual design of the collapsed nodes by embedding simple glyphs that indicate the topo-

logical patterns of subgraphs induced by clusters. Visual smoothness is achieved differently for animations and timelines. For timelines, we link identical objects across consecutive time steps so as to obtain a fluent tracking of the evolution of a certain object. In animations, frames are played in sequence, which makes it hard to maintain the mental map and compare different snapshots. Hence, we improve their visual stability from two aspects. One is the graph layout. We build a Super Graph by aggregating all snapshots and then compute node positions by applying the force-directed algorithm at a global level. The other aspect is the design of glyphs that denote the topological patterns. Visual changes caused by the inter-transformation of glyphs are minimized by adopting a pentagonal design. In one frame, we place previous and current glyphs concentrically to help users do comparisons.

1.3 Thesis Structure

The remaining part of this thesis is organized as follows. In Chapter 2, we carry out a survey on previous work related to dynamic graph visualizations. In Chapter 3, we introduce how to visualize the similarities between graph communities based on a comprehensive framework. Chapter 4 is about representing large dynamic graphs by smooth and simplified animations. In Chapter 5, we describe modeling the relationship between geo-textual data as time-varying graphs and carrying out urban analysis on such graphs. The thesis ends with Chapter 6 that covers the conclusions and the future work.

CHAPTER 2

LITERATURE REVIEW

It is difficult to visualize dynamic graphs because we need to consider a lot of requirements, from the perspective of not only aesthetics but the effectiveness in analysis. Frequently asked questions include: in what forms should the graph nodes and their relationship be presented? how to project the temporal dimension to the visual space? what are the effective ways to navigate users through complex graphs? To find answers to these questions, we review existing development on illustrating static graphs, revealing evolutionary patterns and extending the visual scalability, as well as the techniques for evaluating the effectiveness of visualizations.

2.1 Requirements and Methods for Visualizing Dynamic Graphs

Aesthetic Criteria

A good visual design should be beautiful and attractive so as to evoke users' interests in exploring the data. Meanwhile, it also needs to be effective to raise the speed and accuracy of exploration [13]. Lau and Moere [14] build a model that takes *data focus*

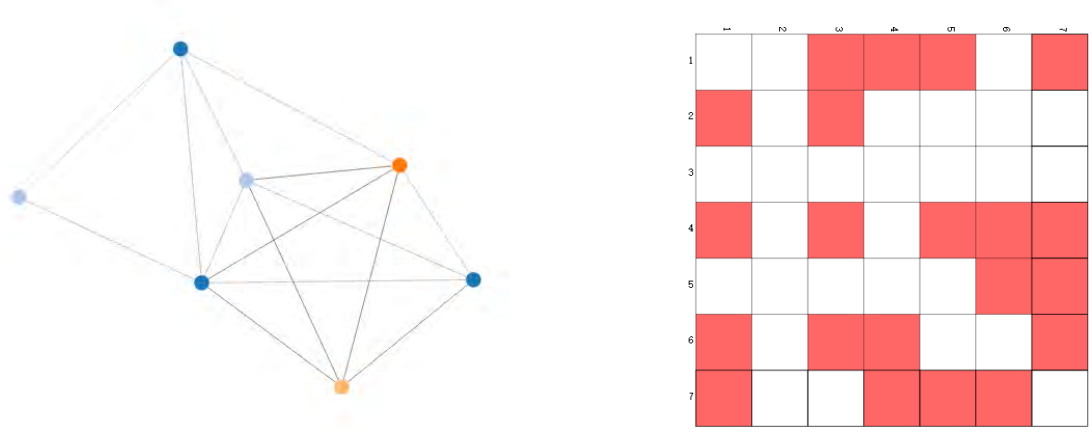
and *mapping technique* as the two key factors of aesthetic criteria. They judge the quality of aesthetics according to the purpose of applications: acquiring knowledge or designing visual mappings. Elements that are frequently used for mapping include *color*, *position*, *size*, *typography*, *conveyance* and *interaction* [15]. They often work together to enhance the functionality and reliability of visualizations [16].

The primary decision of drawing graphs is what glyphs to use to represent graph entities, i.e., vertices and their connections. Then, we should consider where to place them on the display. Given a dynamic graph, another step of graph drawing is to reduce the visual movement of identical entities across time so as to obtain a stable result. According to Beck et al [17], aesthetic issues are raised by following aspects:

- *Graph drawings*: representations such as the node-link and matrix diagrams shown in Figure 2.1 have different properties, and thus cause various issues. With the growth of graph size, node-link diagrams may suffer from clutter due to the limited space of display. It will be difficult for users to recognize entities. Well-designed layout algorithms might relieve the problem. By using the matrix representations, we do not need to worry about the graph layout. Matrix representations are suitable for dense graphs and good at solving problems like finding the shortest path [18]. An important requirement for drawing matrices is to increase the visual compactness.

- *Temporal representations*: the requirement is to preserve users' mental map [19]. Namely, the visual outlook of the graph is not supposed to change suddenly between two subsequent time steps. Otherwise, tasks like finding differences become

impossible because of too many varying elements. To do so, visual properties such as shapes and positions should not change considerably, especially for identical graph entities.



(a) Node-Link diagram.

(b) Adjacency Matrix.

Figure 2.1: A node-link diagram explicitly shows the graph topology and in the corresponding matrix representation, intersection entries are colored if the end nodes are connected.

Researches on aesthetics often focus on graph-theoretical properties and rarely consider the semantics associated with the data [20]. They aim to obtain pleasant layout generated by complying with heuristics that have perceptual basis such as the Gestalt principles and Norman's emotional design framework [21]. Based on node-link diagrams, aesthetic requirements include [22]:

- Minimize edge crossings: if there are too many intersections of edges, it will put a lot visual burden on users. They can hardly identify a specific edge, and crossings may even overlap nodes.

- Minimize node and edge overlap: minimize ambiguity by avoiding over-clustered nodes or edges.
- Maximize symmetry: a symmetric layout can help users to understand the graph structure better.
- Uniform edge lengths: they might be helpful to prevent the graph being distorted.
- Uniform node distribution: alleviate the visual clutter problem to some extent.
- Separate non-adjacent nodes: separate nodes by their connectivity, and adjacent nodes appear to be more related.

An ideal layout algorithm is supposed to satisfy these requirements as many as possible. However, due to the competitive nature of the aesthetic criteria, we have to achieve a balance between them in certain contexts.

Graph Layout

Graph layout algorithms calculate the node positions on the display according to the aforementioned aesthetic metrics, or based on the information that we want to highlight.

Currently, the force-directed [23] algorithms are widely adopted. The idea is to treat the graph like a physical system where simulated forces are assigned to nodes and links. Attractive forces pull endpoints of edges together, while repulsive forces push all pairs of nodes apart. Forces can be analogies to different physical behaviors like springs and

particles [24], or gravity [25]. The ultimate goal is to make edges have approximately equal lengths and to minimize crossings between them.

P. Eades [26] create the pioneering work in force-directed layout. A spring-electrical system is used to achieve a balance state where each node bears zero-force. Based on Eades' approach, another type of method aims at minimizing the energy by solving a cost function. Kamada and Kawai [27] assumes that the Euclidean distances between nodes should approximate their graph-theoretic distances. The cost function is the squared difference between these two. Indeed, it implies the extend of asymmetry. The difference between energy-based algorithms is that they use different strategies to establish and solve objective functions. They are likely to get stuck in local optima. Figure 2.2 displays the Les Misérables dataset, using variants of the force-directed algorithm.

(a) (b) (c)

Figure 2.2: Graph layouts generated by three force-directed layout algorithms. (a) ForceAtlas algorithm tends to keep the nodes that have higher degrees in the center. (b) Frunchterman's algorithm always keeps the symmetrical sphere shape. (c) Yifan Hu's method pushes nodes with low link counts to the periphery.

Eades' algorithm and its variants suffer from scalability issues. Poor layouts are gen-

erated and time costs become unacceptable, when dealing with graphs of massive nodes and dense structures. The GEM (Graph Embedder) [28] algorithm introduces a gravitational force to pull nodes to the barycenter of their neighborhoods. It runs faster and is able to handle larger graphs. Multi-level algorithms [29] are also used to improve the efficiency. They partition graph nodes into groups, then conduct layout algorithms at the coarsest level of groups. Finally, the layout is propagated iteratively back to the original graph. Multi-level algorithms are different at the coarsening schemes. For example, Topo-Layout [30] detects topological features, such as trees and complete graphs and collapses these components into a single node.

Special layout strategies are available for other graph drawing techniques. Spectral-based drawings use eigenvectors of the adjacency or laplacian matrix to compute node coordinates. Tree layouts are suitable for graphs with tree structures. Arc diagrams place nodes along an axis and connect them with curves. Algorithms designed to serve extreme conditions are also in use. For example, the space filling curves generate layouts with no overlapping between nodes [31].

Graph layout can be affected by the properties of entities. Bezerianos et al. build a two-view visual system as shown in Figure 2.3 for analysing multivariate graphs [32]. One view displays a scatter matrix which provides exhaustive combinations of pairs of attributes. The other view represents different graph layouts when any two attributes are selected. Hosobe [33] computes high-dimensional layouts by eigenvalue-based scaling techniques. Then, 2D layouts that are adaptive to user interactions are obtained by satis-

fying certain constraints.

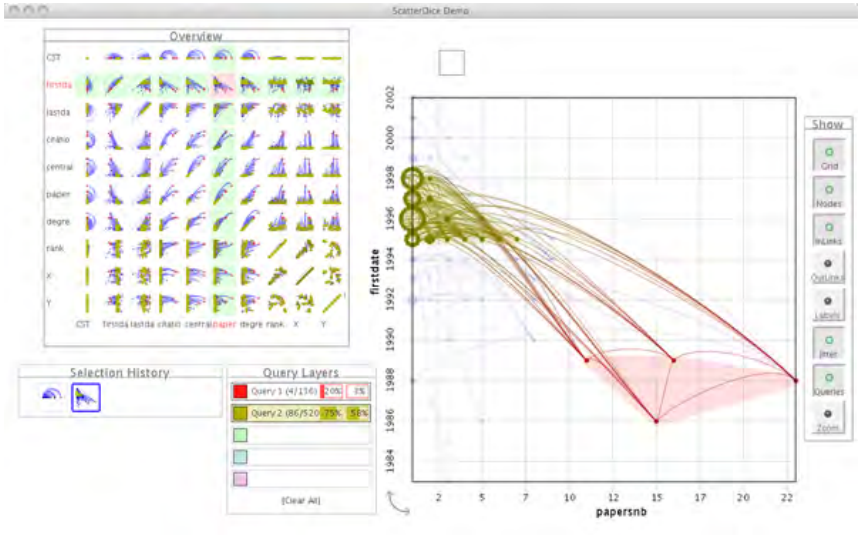


Figure 2.3: Use GraphDice to visualize the co-author dataset in InfoVis 2004 Contest.

Timeline Versus Animation

There are two streams of approaches to depicting the temporal dimension of dynamic graphs, one is animation and the other is timeline that contains a sequence of small multiples.

Animation

The evolution time of dynamic graphs is mapped to the visual time of animations [34]. An important task is to preserve the mental map that refers to a user's abstraction of structural information when looking at the graph layout [35]. A stable layout helps users to acquire a consistent understanding of the graph evolution. To achieve the goal, visual

differences between consecutive frames of the animation should be minimized. Frishman et al. [36] proposed an incremental method of drawing dynamic clustered graphs. It aims at maintaining the size and position of clusters as much as possible. They use invisible occupiers to save space for new nodes so that the animation will not be interrupted by sudden changes.

Animations can be categorized into *online* and *offline* techniques, depending on the available time steps:

- *Online*: only past time steps are available. Gorochowski et al. keep local structures relatively stable by using the concept of aging [37]. Lee et al. [38] implement a simulated annealing with a cost function assembling a group of weighted drawing criteria. The optimization result then helps to derive a stable layout.

- *Offline*: both past and future time steps are considered. Feng et al. [39] achieve the spatiotemporal coherence through the concept of *Super Graph*. Individual time steps are assigned an importance score to control the playing speed of animation. Diehl et al. [40] focus on node groups and reuse group positions to achieve stability.

Generally, online animations are more flexible as they do not necessarily require for the knowledge of the entire evolution. Hence, they are suitable for real-time applications. By contrast, offline approaches tend to generate optimal layouts and preserve the visual stability better.

Timeline

The time dimension is mapped to the space in timelines. Snapshots at individual time steps are sequentially displayed in one view. Timelines make it convenient to compare different snapshots. Users do not need to keep previous steps in mind. However, due to the limited space of display, timelines are criticized for the poor scalability of showing an increasing number of time steps.

Visual consistency can be maintained differently according to how graphs are presented in snapshots. For node-link diagrams, positions of identical nodes are expected to keep unchanged over time, as shown in Figure 2.4. In Figure 2.5 (a), nodes are stacked along vertical axes denoting timestamps and edges are drawn between two neighboring axes [41]. Similarly, edges in Figure 2.5 (b) are drawn as curves at individual time points [42]. Axis-based representations get node positions fixed, but they do not cater to graphs where nodes come and go frequently because of the low utility of space.

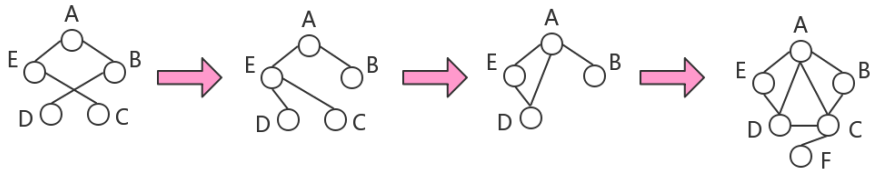
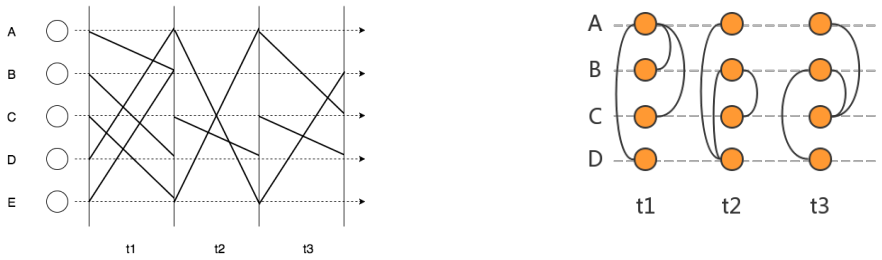


Figure 2.4: In the timeline of node-link diagrams, the positions of identical nodes at consecutive snapshots are supposed to keep the same to achieve the best visual stability.



(a) (b)

Figure 2.5: Stacking nodes along the vertical axes to improve the visual stability. The vertical positions of nodes are fixed, and edges exist: (a) during a period; (b) at certain time steps.

The timeline itself can be drawn in different shapes. The most widely adopted is a straight line with the reading direction from left to right. In Figure 2.6, TimelineJS marks timelines to show the occurrences of events.

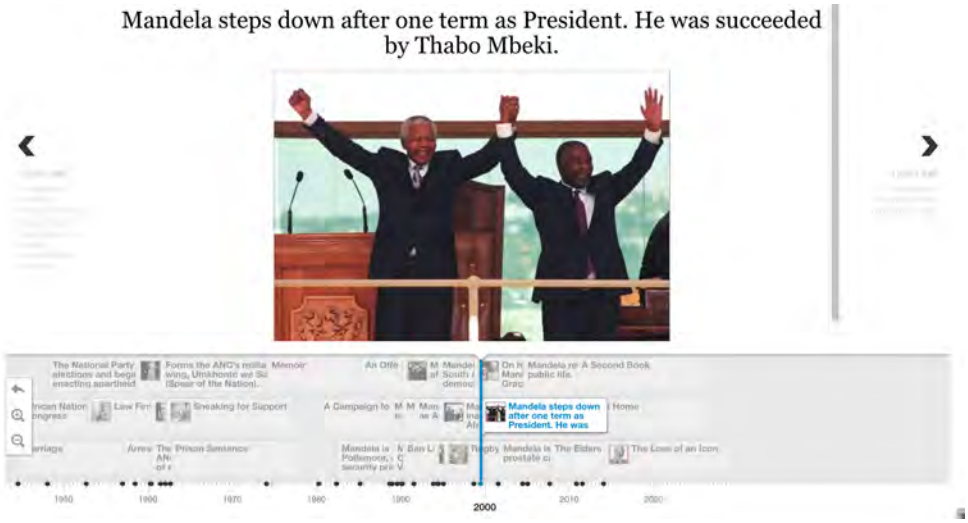


Figure 2.6: Use TimelineJS to demonstrate the life of Mandela. Each life event is further explained in a slide-show styled figure[1].

Non-linear timelines can be radial, spiral and even arbitrary lines [43]. Radial timelines reflect the periodic nature of time. RJ Andrews use them to reveal how celebrities spent their daily time [44], as shown in Figure 2.7. It is straightforward to find the habit of individual celebrities. A high space-filling approach is to draw timelines as spirals [45] that are both aesthetic and suitable for displaying dense time events. Arbitrary representations are similar to spirals except that their shapes are arbitrary instead of spinning around a center [46]. It sometimes causes heavy visual burdens, because their reading directions may change back and forth.

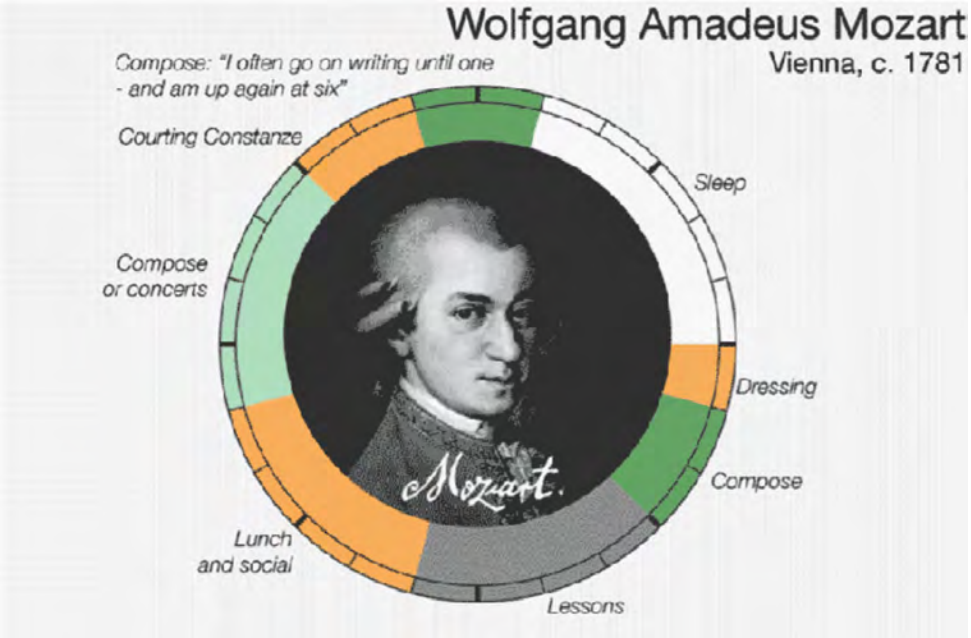


Figure 2.7: The radial timeline to represent the daily routine of a celebrity. Each time period is accompanied by some explanation words[2].

[1]https://timeline.knightlab.com/.

[2]http://www.infowetrust.com/navigating-data/.

Besides the line-based representations, grid diagrams can also be used to display multiple snapshots. One typical example is calendar [47]. As the interval between cells has uniform granularity, such as day, month, year, space will be largely wasted if the data is not evenly distributed. To deal with the uneven distribution of data items, non-chronological time scales need to be used.

We compare the foregoing representations in Table 2.1. Generally, straight lines and grid diagrams are convenient for observation, and radial and spiral timelines are more space-efficient. We can use arbitrary shapes to expose representative states by placing similar snapshots closer [48].

Representation	**Advantages**	**Disadvantages**
Linear	intuitive perception	low scalability
Radial	periodic time	low space-filling
Spiral	high space-filling, scalable	difficult to compare
Arbitrary	flexible	mussy reading directions

Table 2.1: Comparisons between different shapes of timelines.

In case that multiple timelines appear in one view, we need to align them to expose significant patterns. Individual timelines may depict different categorical attributes like topics of news [49], different streams of events [50]. Or, each timeline corresponds to a temporal division, e.g., year and month [51]. The juxtaposed placement of timelines makes it convenient to detect concurrency and overlaps. To show transitions between different timelines, animation techniques might be helpful [52].

Compound

Animations and timelines both have pros and cons. The selection criteria should be based on data types and the requirements of applications. We prefer animations to show graphs that expand over a long time span. However, complicated exploration tasks are better served by timelines [53]. In some circumstances, it is better to combine the two techniques and exploit their strength.

Hadlak et al. provide an approach that adapts the visualization locally so that different facets of data can be displayed [54]. The system conforms to an iterative loop of analysis and allows users to select a subgraph so as to embed different visualizations. DiffAni aggregates parts of the timeline into animations [55]. It consists of three tiles. *Diff-tiles* show difference maps over certain time period, *animation-tiles* represent the graph evolution, and *small-multiple-tiles* display graph states at specific time points. Based on the edge-splatting technique [41], Beck et al. focus on animating timelines to improve the scalability of visualization with respect to time steps [56].

Interaction

Interaction and navigation facilities are essential in visualizations, because the layout algorithms cannot solve all problems caused by the large size and complex structures of graphs [57]. Interaction techniques can provide users with multiple levels of details and increase the visual scalability to some extent. Navigation tools help to locate entities quickly by different rules. Commonly used interaction and navigation techniques are as

follows [58]:

- Overview: provides a sketch of data without showing details. It allows users to have an overall concept of the objects and helps them to decide which targets are of interest;
- Zooming: includes zoom-in and out. The former supports browsing details in high resolutions and the later fits data of larger scales and thus extends the scalability of visualizations;
- Selection: reduces the number of entities to be displayed. It often implements by complying with a few filtering rules;
- Highlighting: distinguishes important patterns or phenomena from less important ones. It attracts peoples' attentions through special colors, shapes and so on;
- Brushing: interactively selects data from a visual representation. Users are allowed to apply the operation multiple times, and the brushed parts can be used for comparison;
- Focus+Context: deforms the visualization by enlarging the component of interest while contracting the remaining part. Meanwhile, the context information is preserved;
- Incremental Exploration: displays a small portion of data and the rest are available as needed. To explore the whole set, the visible window slides along a predefined path.

It remains an open question of how to effectively couple diverse interaction tools. Keefe and Isenberg [59] raise six challenges that have to be solved in order to achieve a good combination of interactions and natural human-computer interfaces. Integrating more interactive features into the system does not mean that a higher usability can be achieved. One practical way is to optimize the combination by collecting feedbacks from users.

2.2 Simplifying the Representation of Large Graphs

With the increase of graph size, we need solutions like the abstraction and compression to reduce the complexity of visualizations without affecting the exposure of significant patterns [60]. Because placing an enormous number of nodes and edges in one view either exceeds the limit of the screen or causes severe clutter problems for both animations and timelines. Existing techniques can be categorized by whether they manipulate the original data (e.g., sampling, clustering) or the visual encodings (e.g., edge bundling).

Graph Sampling

Sampling aims at decreasing the number of elements that need to be presented to a manageable size. Techniques can be categorized according to the targets to sample, i.e., nodes, edges and paths. Representative methods [61] are:

- Random node sampling: randomly select nodes with uniform probability. The sampling result is the subgraph induced by the set of selected nodes;
- Random edge sampling: randomly select edges with uniform probability. The sampling result is the subgraph induced by the set of selected edges;
- Random walk sampling: randomly select a node at the beginning, and iteratively select the next node from neighbors of the current node until reaches the sampling.

We evaluate these techniques by validating the preservation of structural properties, such as the degree distribution and clustering coefficient. Leskovec and Faloutsos claim that 15% sample is usually enough to match these properties with the original graph [61]. When investigating the impact that sampling has on visualization results, Nguyen et al. [62] put forward a list of metrics to measure the visual quality. Wu et al. [63] found that different sampling strategies preserve different visual features.

Node Clustering

Representing a cluster of similar nodes by a new single node may profoundly reduce the number of visual elements. We can also gain insights at a higher level. In fact, the incident edges are also clustered. Here we mainly consider methods that are based on measuring the node similarity.

Spectral methods take the laplacian matrix as the input and calculate eigenvectors that each denote a graph node. Using a traditional clustering technique like k-means, nodes

are clustered based on similarities computed in a vector space [64]. Before using such kind of method, we have to decide the number of clusters so that nodes within the same cluster are mostly related.

Community detection techniques do not require prior knowledge on the number of clusters. They return natural node groups with dense intra-connections and sparse inter-connections. We call these groups communities, and it is essentially an optimization problem to find them. Different implementations have their own objective functions and then apply approximation algorithms or heuristics to solve the function [65].

There have been several algorithms that prove to have a good performance [66]. The algorithm invented by Blondel et al. [67] aims at maximizing the *modularity*. Modularity is a scalar value which measures the density of links inside communities as compared to links between communities. The algorithm executes iteratively until modularity does not increase any more. The running time is linear to the number of links. The Girvan–Newman algorithm repeatedly removes edges with highest betweenness centrality until only communities are left. It takes $O(nm^2)$ on a graph with n vertices and m edges [68]. Rosvall and Bergstrom [69] convert the community detection into an information compressing problem. The key point is that one can always recover as much as possible the graph structure when decoding the compressed information. The quality function is solved by combining the greedy search and simulated annealing [70]. The algorithm proposed by Ronhovde et al. minimizes the Hamiltonian of a Potts-like spin model. The spin state denotes the membership of a node in a specific community. Com-

munity scales are allowed to span according to the value of a resolution parameter. This algorithm runs fast with a super-linear complexity in the number of graph links. Fortunato and Barthelemy [71] found that modularity-based approaches may always fail to detect communities smaller than certain scales. Hence, they suggest to check the structure of all detected modules. Based on a concept called *resolution limit*, they make a trade-off between the number of modules and the value of terms.

A node may belong to multiple communities in certain cases, which is a common phenomenon in social networks. Approaches for detecting such *overlapping communities* [72] include clique percolation [73], line graph and link partitioning [74], local expansion and optimization[75], fuzzy detection[76], and agent-based and dynamical algorithms[77].

Multi-level or hierarchical methods are helpful to control the extent of clustering. With the growth of hierarchy, a graph is more deeply clustered. Clémençon et al. suggest that we should validate the significance of clusters by comparing their modularities with that of a random graph [78].

Edge Bundling

The foregoing methods tend to realize visual simplification by reducing the number of elements to be represented. However, edge bundling directly operates on node-link diagrams and returns optimized results. The goal is to remove visual clutter as much as possible and reveal high-level edge patterns. Close edges are deformed and are bound in groups.

Bundling techniques can be geometry-based, cost-based and image-based [79]. Geometries like trees [80] and grids are used to decide the shape of edges. Grids can be uniform or non-uniform. Triangulation is a typical way to produce a non-uniform mesh [81]. Intersections of mesh edges and graph edges are control points that segment graph edges into connected paths. Edge bundling is then converted to a procedure of merging control points. However, curvatures along the bundling direction might vary a lot. To minimize the variation, Holten et al. model edges as springs that have attractive forces between each other [82]. The energy generated by the simulated system is minimized to obtain a bundling result. Ink used to draw edges can also be taken as a cost to minimize [83].

To summarize, we have reviewed the methods that facilitate the visualization of large graphs by reducing or aggregating visual elements. Other techniques may also serve the purpose of visual simplification. Landesberger et al. aggregate mobility graphs spatially and temporally [84]. Instead of showing all snapshots, they cluster similar graphs and leave a few representatives to demonstrate the evolution of human mobility. Dwyer et al. use a module to replace a set of nodes that have common neighbours, then a link connected to the module implies connections to all members inside [85]. Another way to group nodes is to inspect their membership of a particular topology like a clique. Then, visual complexity is reduced by using specially designed motifs to substitute the nodes that contribute to the same topology [86].

2.3 Revealing the Evolution of Dynamic Graphs

The objective is to explore the trend of graph changes. For simple graphs, changes include the insertion and deletion of nodes and links. For attributed graphs, changes can also be the increase and decrease of attribute values. It is infeasible to visualize all low-level changes becuase the obtained visualizations will overwhelm users and make it difficult to find useful information. Therefore, we focus on changes that occur to mesoscopic objects such as communities.

2.3.1 Dynamic Community Detection

Regarding dynamic networks, we need to expose the consistency between communities that are detected at consecutive time steps. According to Rossetti et al. [87], there are three types of methods:

- *Instant Optimal*: calculate communities independently for each graph. Connections between communities at consecutive time steps are decided by their overlaps;
- *Temporal Trade-Off*: communities detected at the next time step depend on communities found at last timestamp;
- *Cross-Time*: consider graphs at all time steps and compute communities at a global level.

Shang et al. [88] put forward a real-time algorithm aiming at tracking communities on

a fine-grained level. The algorithm takes the result of the Louvain method [67] at the first time stamp as an initial input, and updates it at following time stamps. Increased edges are classified into four types, each corresponding to different operations on previous communities, including keeping unchanged, combining and so on. However, the algorithm only deals with frequent and incremental changes and lacks the processing of the decrease of edges. By optimizing multiple objectives simultaneously with genetic algorithms, Folino et al. get communities that trade off between the clustering accuracy and the deviation between time steps [89]. In terms of streaming data, strategies of locally updating communities that are affected by adding or removing nodes or links are adopted [90]. The objective is to smooth the evolution of communities while avoiding external matching. Accordingly, a label propagation procedure broadcasts the changes to the neighbors of the node and adjust the local community memberships. This method largely decreases the running time of dynamic community detection. Zakrzewska and Bader dynamically expand the seed set when graph changes and the community that each seed belongs to is updated. The algorithm also supports parallel processing. Gauvin et al. [91] factorize tensors that take time as one dimension. Graph structures are represented by adjacency matrices. The temporal activity patterns of communities can be extracted after factorization.

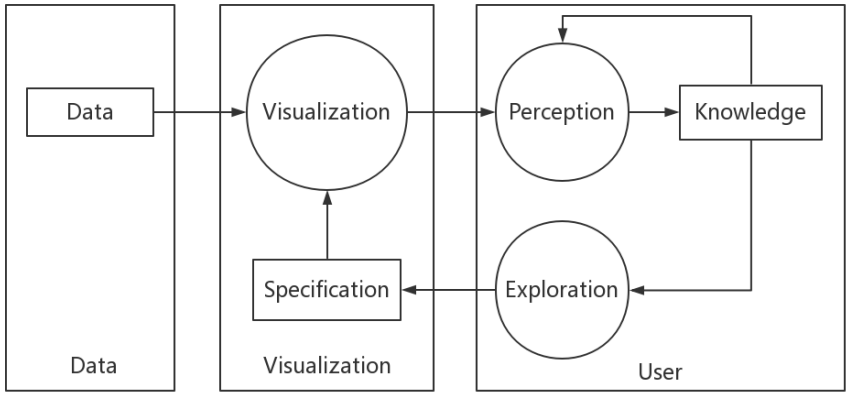


Figure 2.8: Visual analytics involve iterative acquisitions of knowledge from the perception [94].

2.3.2 Analytic Tasks of Graph Evolution

Visual analytics can be regarded as a supporting process for decision making. Traditional visualization researches do not necessarily deal with visual tasks nor use advanced data analysis algorithms [92]. The objective of analysing graph evolution is to track the changing trend of graph structures and properties. Studies on graph structures focus on both the overall scale and the local components [93]. For many real-world networks, vertices and edges have intrinsic attributes. Analysing such attributes may promote applications like discovering influential entities and predicting the occurrence of events.

A generic process of visual analytics [94] is illustrated in Figure2.8. The loop denotes the iteration of knowledge discovery until we obtain the maximal information.

Exploration procedures are driven by a set of well-designed tasks. Common tasks include: viewing, filtering, zooming, manipulation, querying and so on. Visual tasks

can be categorized on different facets such as data types, transformation operators and function queries. After inspecting 53 visual systems, Ahn et al. come up with a more comprehensive approach for classifying tasks [95] based on three dimensions:

- *Entities*: not only refer to nodes and links but also groups and the whole graph. Sometimes, users are interested in node or link groups such as clusters and communities [96]. The groups can be generated by either position or attribute information;
- *Properties*: include the structural property and the domain property. Structural property reflect the topology information of the graph, which may include degree, centrality and so on [97]. Domain properties add up the complexity of analyzing the dynamic graph, because usually domain properties need to be correlated with the structural properties;
- *Temporal features*: indicate the way for users to observe and compare the states of entities and properties over time. State transformation of the graph is caused by different types of events. These events can be simple addition or deletion of nodes and edges at a specific time. Or they can be the aggregation of simple events over a period. Aggregated events uncover the change of properties and the change itself can be demonstrated qualitatively (stable, convergent, grow) or quantitatively (speed, acceleration) [98].

One frequently executed task is comparing graph snapshots so as to find changes that happen at different time steps, or to summarize graph states during a period. Abello

et al. [99] detect graph discrepancies by marking each edge with a sequence of time labels when two endpoints communicated, and turn the time-varying graph into a collection of time-stamped communication pairs based on the concept of set-system. The key component in their algorithm is using combinatorial discrepancy [100] to isolate characteristic patterns from time-varying graphs. Nesbitt and Friedrich applied the Gestalt Principle [101] to animations of dynamic graphs to help users understand the changes that happened in the graph layouts [102]. The Gestalt Principle considers several aspects about how peoples' perceptual systems organize disjoint visual elements into groups. GraphDiaies [103] was designed for identifying, tracking and understanding changes happened in animated node-link graphs. Exploration tasks help in time reducing and error minimization by supporting multiple navigation features, such as inter and intra navigation between and within staged transitions and the non-linear navigation over time. Wongsuphasawat and Shneiderman applied a Match & Mismatch method to categorical data so that users can effectively find similar items [104]. Before matching and comparing entities, users are allowed to align the sentinel categories.

2.3.3 Visualizing Structure and Property Changes

To show the structural changes of communities, Reda et al. create interactive visualizations [105]. Individuals are depicted as threads that enter different communities to show the temporal change of memberships. Similarly, Vehlow et al. [106] represent the dynamic communities together with the topology of original graphs as references. They provide a

SankyFlow-liked diagram with optimized ordering of vertices, so that users can intuitively trace the movement of vertices across communities. An example of SankeyFlow is given in Figure 2.9. Wu et al. analyse time-varying egocentric graphs and build a three-layer interface for displaying patterns [107]. Users can have an overview of the entire graph dataset at the macroscopic level and the evolution of an individual researcher's egocentric graph is accessible at the mesoscopic level. The microscopic level reveals the temporal information of egos and alters.

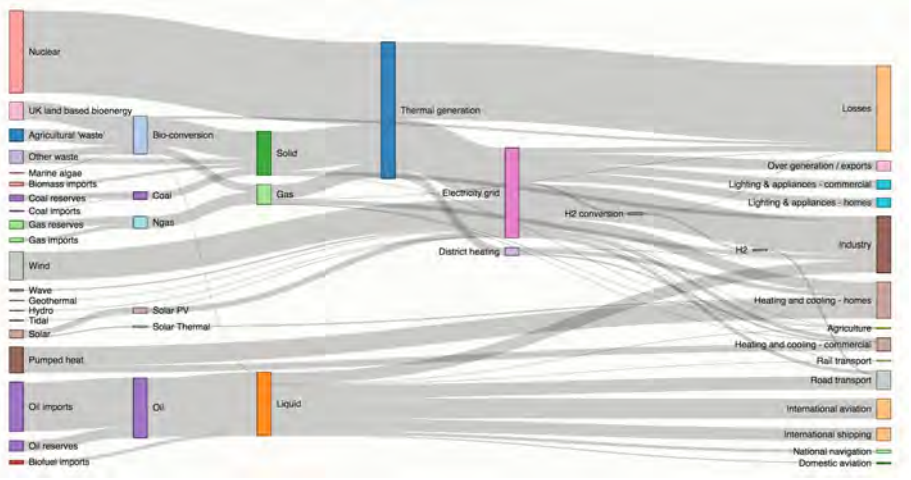


Figure 2.9: The *Sankey* diagram[3]is suitable for showing the flow of data proportions.

Based on the properties of graph entities, Yuru et al. find dynamic multi-relational clusters [108]. They firstly generate soft clusters by picking important entities and computing their similarities. The clusters are measured by calculating the probability of an entity transiting from one cluster to another. Their method acquires a good performance in tracking evolution and revealing contextual transitions. EgoNetCloud [109] is imple-

[3]https://bost.ocks.org/mike/sankey/.

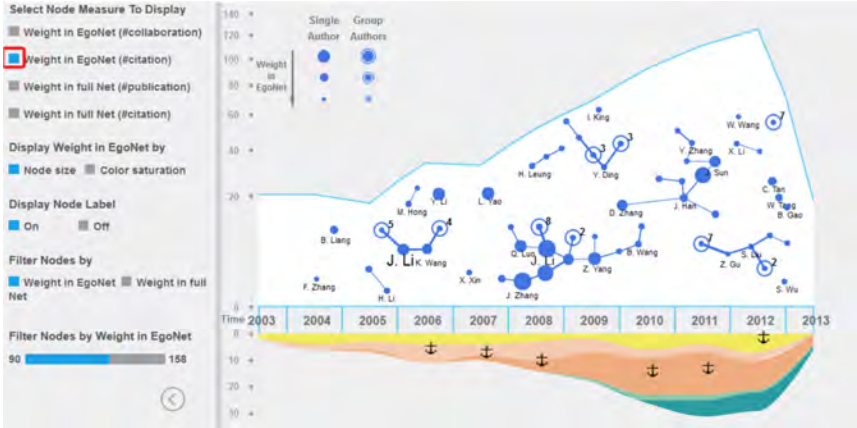


Figure 2.10: Use EgoNetCloud to show the co-authorship of a researcher from year 2003 to year 2013 [109]. Multiple interaction features are provided in the left panel.

mented to support the exploration of event-based egocentric graphs. Data-driven simplifications are applied to obtain small but salient abstractions of large graphs. A novel layout algorithm constraints the positions of nodes, so that similar nodes are placed closely with less visual clutter. Figure 2.10 is an example showing the evolution of co-authorship egocentric graphs.

2.4 Evaluation Methods

We evaluate a visual system to prove its effectiveness in helping users to gain insights into the data. The general procedure of evaluation can be concluded as follows:

1. Clarify the goals of evaluation. For example, evaluating the performance of a visual system with respect to a specific data set;

2. Determine the metrics for evaluation. The metrics can be qualitative or quantitative.

Details of these two types of metrics are listed separately in following parts;

3. Decide the approaches of evaluation. This process should include at least two components, choosing testers (whether they are experts in one area or ordinary users) and selecting the form of evaluation (e.g., interview or questionnaire);

4. Design a questionnaire. List all the questions that are required to be answered by target users. The questions are supposed to cover all aspects of the visualization system as much as possible;

5. Demonstrate and analyze the results of evaluation. With the combination of quantitative measurements and feedback from target users, researchers can draw a conclusion about which part of the system has been relatively desirable, while other parts may need improvement.

The targets that we evaluate are mainly: the methods for analysing data and the visual designs for presenting the results of analysis. We can justify the evaluation by setting a ground truth which needs to work with visual tasks [110].

Evaluation metrics are categorised as objective or subjective. The former can be described quantitatively while the latter are usually derived from users' experience. Based on the metrics proposed by Frishman and Tal [36] for assessing the performance of the dynamic drawings of clustered graphs, we make complements, and list them as follows:

Quantitative:

- Space Compactness: it measures the utilization of the display space. This is an important measurement because with the fast growth of data size, how to display as much information as possible in limited space becomes a primary issue;

- Run-time Efficiency: here we can take the graph layout algorithms as an example. Most of the force-directed algorithms are relatively slow due to large numbers of iterations. Hence, they can hardly be applied to large scaled datasets;

- Error Rate: it usually comes along with visual tasks. If a benchmark is available, we can compare it with the result of tasks to calculate the error rate. Lower error rates partially reflect the effectiveness of visualization tools;

- Frame Changes: changes between two adjacent time steps should not be sharp. Otherwise, users' mental map will be damaged. The number of nodes/edges added to or removed from view, as well as the total movement of nodes can be used to measure the changes;

- The number of edge crossings: it is a unique measurement for graph data. Fewer edges crossings are often preferred, because the clutter caused by edge crossings largely reduces the readability;

- Total edge length: related nodes are expected to be placed as close as possible, so that users' eyesight does not need to move across a long distance.

Qualitative:

- Readability: affects how easily readers can understand the information that the system tries to convey. It is often reflected by the aesthetic criteria as well as the complexity of drawings. Therefore, this measurement is closely related to metrics of edge crossings and edge lengths mentioned above;

- Stability: it shows the extend of mental map preservation. High stability might save users from a sudden visual change, such as a noticeable movement of the node;

- Usability: the visual system should be easy to interact with and manipulate so that even non-expert users can quickly get started.

Currently, most of the evaluation researches focus on controlled experiments, usability assessments and case studies [111]. For any applications, it is desirable to incorporate both the qualitative and the quantitative metrics. We can improve the system by getting users involved and learning from their feedbacks.

CHAPTER 3

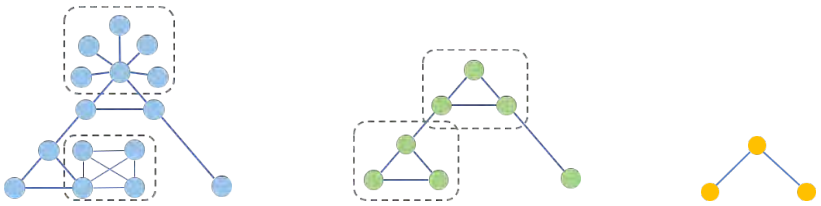
REPRESENTING THE SIMILARITY BETWEEN GRAPH COMMUNITIES

3.1 Introduction

As the number of graph entities increases, the readability of presenting graphs in a limited space decreases. Because it is difficult to avoid overlaps by isolating nodes, or to reduce crossings by organising edges [60]. Consequently, users are hindered to perceive significant information from graph drawings, such as the connectivity patterns.

To facilitate the visual understanding, one solution is to cluster nodes and links and represent them by a new single node [112, 113]. The benefits of doing this include two aspects: (1) elements that need to be presented on display become less, hence the visual clutter is alleviated; (2) we can expose the graph structure at a higher level, as shown in Figure 3.1, the level can be controlled by applying clustering operations iteratively.

In network analysis, the *community structure* is built by a cluster of nodes and their incident edges. Nodes inside the same community are more densely connected than nodes that reside in different communities. Similar communities can have various applications. For example, in the context of social networks, we can carry out friendship recommendations, evaluate emotions, detect relationships, and predict events [114, 115, 116].



(a) (b) (c)

Figure 3.1: Folding nodes and links at different levels. From (a) to (c), we lower the visual complexity by hiding more details from view.

We refer to the graph whose nodes represent communities as the *community graph*. It raises two issues: (1) users can no longer visually access the inner structures of communities; (2) the graph drawing is not informative. From Figure 3.1c, users can count the communities and observe their relationship, but they are unable to recognize the topology of graphs that are induced by communities, neither to tell if the graphs have similar structures.

In this chapter, we put forward a framework to enrich *community graphs* with the structural similarity information among communities. As shown in Figure 3.2, our framework consists of three main modules. (a) and (b) are for detecting communities and calculating structural similarities, respectively. They both incorporate multiple state-of-art approaches, and users can select any of them according to their performance on different datasets. (c) is for drawing community graphs by node-link diagrams. Meanwhile, nodes' positions are decided by community similarities. Details of these modules can be found in Section 3.2, Section 3.3 and Section 3.4, respectively. We further conduct two case studies on real data by following the pipeline of the framework.

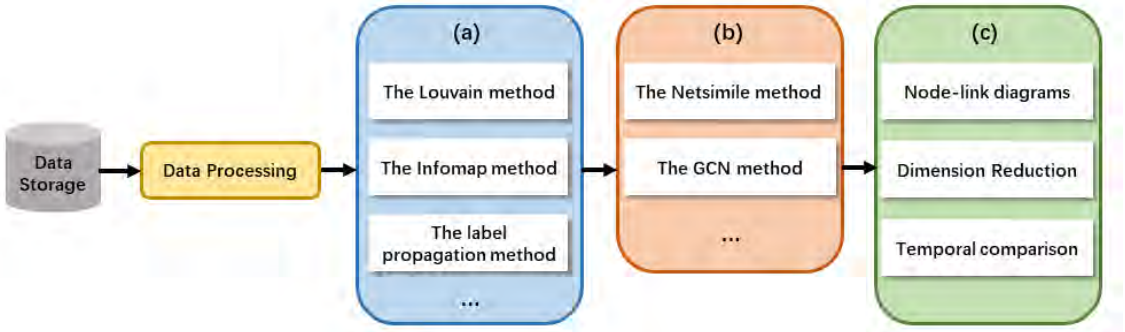


Figure 3.2: The framework of visualizing the structural similarities between network communities.

3.2 Community Detection

Methods for detecting communities from graphs have been extensively studied [117]. In this section, we introduce three of them that are widely accepted for the following reasons: first, they achieve good performance and high efficiency on large complex networks; second, they do not require prior knowledge, e.g., the number of communities. These methods can be used interchangeably for obtaining non-overlapping communities in the *Community Detection* module of our framework. The corresponding details can be found in each of the subsequent sections.

3.2.1 The Louvain Algorithm

This algorithm was proposed by Blondel et al. [67]. It aims to greedily maximize the graph modularity:

$$Q = \frac{1}{2m} \sum_{uv} \left[A_{uv} - \frac{k_u k_v}{2m} \right] \delta(c_u, c_v). \tag{3.1}$$

Q is a scale value in the range of $[-1, 1]$, measuring the density of edges inside communities compared to edges between communities. In Equation 3.1, $2m$ is the sum of all edge weights; for nodes u and v,

$$A_{uv} = \begin{cases} 1, & G \text{ is unweighted} \\ w_{uv}, & G \text{ is weighted} \end{cases} \tag{3.2}$$

w_{uv} is the weight of edge uv; k_u and k_v denote the sum of weights of edges attached to u and v, respectively; c_u and c_v indicate the communities of u and v;

$$\delta(\cdot) = \begin{cases} 0, & c_u \neq c_v \\ 1, & c_u = c_v \end{cases} \tag{3.3}$$

To solve the optimization problem, nodes are assigned to their own communities initially. Then, the following steps are repeated until Q converges:

1. For each node, the change in Q is calculated for moving it from its original community to the community of each of its neighbors, and the node is then moved to the community which results in the largest modularity change. If the movement does

not cause any increase of modularity, the node stays in the original community. According to Equation 3.1, for weighted graphs, nodes u and v are more likely to enter the same community if w_{uv} is larger;

2. Nodes of the same community are grouped and are replaced by a new single node. The weight of edges between any two new nodes equals to the sum of weights of the original edges that connect nodes now belonging to two different communities.

The time complexity of this algorithm is in $O(|V| \log |V|)$, and $|V|$ is the number of nodes in the graph.

3.2.2 The Infomap Algorithm

This is an approach based on information theory. A map describes the flow of information across nodes and links [69]. Suppose the flow path is generated by a random walker. The key to this algorithm is encoding the path with the shortest code length. Accordingly, each community is given a unique code and the inner nodes are distinguished by different Huffman codes. Besides, a Huffman code can be reused by nodes belonging to different communities. For each community, a special code is reserved to denote that the walker jumps out of it.

This algorithm maps the objective of community detection to seeking an efficient path code. A better separation of communities results in a shorter code. The average length of

code for each step is defined as:

$$L(M) = qH(\mathcal{Q}) + \sum_{i=1}^{|C|} p^i H(\mathcal{P}^i), \tag{3.4}$$

where q denotes the probability that community codes appear in the path code; p^i equals to the probability that the codes of nodes of community i appear in the path code; $|C|$ is the number of communities; $H(\mathcal{Q})$ and $H(\mathcal{P}^i)$ measure the average length of code for naming all communities and nodes, respectively.

At the beginning, all nodes are initialized as their own communities. Then, in each iteration the visiting order of nodes is determined by a random walk, and the community of a node is updated by assigning it to its neighbor's community which causes the largest decrease in $L(M)$. The algorithm stops until $L(M)$ cannot be further optimized.

The complexity of this method is $O(|E|)$, and $|E|$ is the number of edges in the graph.

3.2.3 The Label Propagation Algorithm

In this algorithm, each node is initialized with a unique label. We then iteratively update the label of each node with the most frequently occurring label in its neighborhood. [77]. At the beginning, since all nodes have different labels and the frequency of each label is 1, we randomly select a neighbor's label to update the label of the current node. The random selection also applies, if there are multiple labels that have the same highest frequency in the neighborhood. The algorithm stops when for every node, it has the same label that most of neighbors have. At the end, nodes that possess the same label are assigned to the

same community.

There are two modes available for updating labels: *synchronous* and *asynchronous*. For the former, a node updates its label based on the labels of its neighbors in last iteration, while for the latter, labels that are accessible to the current node are always the newest. In the asynchronous mode, the updating order of nodes should be random. In this chapter, we use a *semi-synchronous* implementation [118] which determines the visiting order of nodes and the avaiable labels of neighbors by a coloring technique [119]. This implementation is guaranteed to always converge to a stable labeling.

The complexity of label propagation is $O(|E|)$.

3.2.4 Evaluation

It is still an open question to verify if one method is better than others at detecting communities in real applications, because there is no way to know the graph structure in advance. Frequently used benchmarks include the GN (Girvan and Newman) [68] and the LFR (Lancichinetti-Fortunato-Radicchi) [66] datasets. They consist of artificial graphs that are generated by controlling a set of properties [120]. For example, the average degree of the GN network is 16 and the nodes have approximately the same degree. However, these benchmarks have limitations as they cannot take all graph properties into consideration and simulate real networks. In this work, we decide to compare the three methods by applying them on real network data in Section 3.5.

Suppose U and V are two partitioning results of communities. *Normalized Mutual*

Information (NMI) is a metric that can be used to measure the similarity of U and V. It is defined as follows:

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{H(U)H(V)}}, \tag{3.5}$$

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log(\frac{P(i,j)}{P(i)P'(j)}), \tag{3.6}$$

$$H(U) = \sum_{i=1}^{|U|} P(i)\log(P(i)), \tag{3.7}$$

$$H(V) = \sum_{j=1}^{|V|} P'(j)\log(P'(j)), \tag{3.8}$$

$$P(i) = \frac{|U_i|}{N}, \tag{3.9}$$

$$P'(j) = \frac{|V_j|}{N}, \tag{3.10}$$

$$P(i,j) = \frac{|U_i \cap V_j|}{N}. \tag{3.11}$$

The NMI value is in the range of $[0,1]$, and a higher NMI implies that U and V are more similar.

3.3 Measuring Graph Similarities

Graph similarity has a wide range of applications in the area of visualization. For example, Kwon et al. [121] estimate the layout of a graph from the known layout of a similar graph. They adopted a kernel-based method to calculate the graph similarity. Graph kernel can

be defined as a distribution of a set of sub-structures, including random walks, graphlets, the shortest path, etc. Using the persistent homology technique, Hajij et al. [122] quantify structural changes in time-varying graphs, and they can successfully capture cyclic patterns and one-time events.

In this section, we use two vectorization-based methods. Original graphs are transformed into feature vectors and the similarity score is then computed on these vectors.

3.3.1 A Topological Features Based Method

Netsimile [123] is designed to convert graphs into vectors and is independent of graph size. To do so, we need to compute $|F|$ topological features for each node and construct a $|N| \times |F|$ feature matrix which is defined as:

$$M = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1|F|} \\ f_{21} & f_{22} & \cdots & f_{2|F|} \\ \cdots & \cdots & \cdots & \cdots \\ f_{|N|1} & f_{|N|2} & \cdots & f_{|N||F|} \end{bmatrix}, \tag{3.12}$$

where f_{ij} is the jth feature of node i. Features include:

- *Node degree*: the number of immediate neighbors of a node;
- *Clustering coefficient*: the ratio of the number of triangles connected to a node to the number of triples centered at the node;
- *Average number of a node's two-hop away neighbors*;

- *Average clustering coefficient of the immediate neighbors of a node*;
- *Number of edges in a node's egonet*: the egonet is a subgraph induced by the node and its neighbors;
- *Number of outgoing edges from a node's egonet*.

For undirected graphs, we can use less features, discarding the last one.

A feature matrix is then converted to a signature vector V by aggregating the values of each feature:

$$V = [x(M_1), y(M_1), z(M_1), \ldots, x(M_{|F|}), y(M_{|F|}), z(M_{|F|})], \tag{3.13}$$

where M_i, $i \in [1, |F|]$ is the i-th column of M; $x(\cdot)$, $y(\cdot)$ and $z(\cdot)$ represent the median, mean and standard deviation aggregators, respectively.

The run time complexity of Netsimile is linear on the number of edges.

3.3.2 A Graph Convolutional Network Based Method

Node representations can also be acquired from learning the whole graph by a generalized neural network model. The benefit is that we do not have to designate the appropriate features to compute for each node. The GCN (Graph Convolutional Network) model extracts graph features by conducting convolutions in the spectral domain [124].

We adopt an implemention proposed in [125]. The netwok structure is demonstrated in Figure 3.3. By training this network, we aim to obtain node-level outputs of a function

of the features of the input graph. In this context, the final outputs are labels that separate nodes in to c classes. Graph features are represented as a $|N| \times |N|$ matrix. Each row is the feature vector of a node that shows its *one-hot* occurrence. In every convolutional layer, node features are updated, and the feature dimension at the last layer equals to c. After applying the *softmax* operator to the output features, we can get the distributions of probabilities that nodes are assigned to each type of classes.

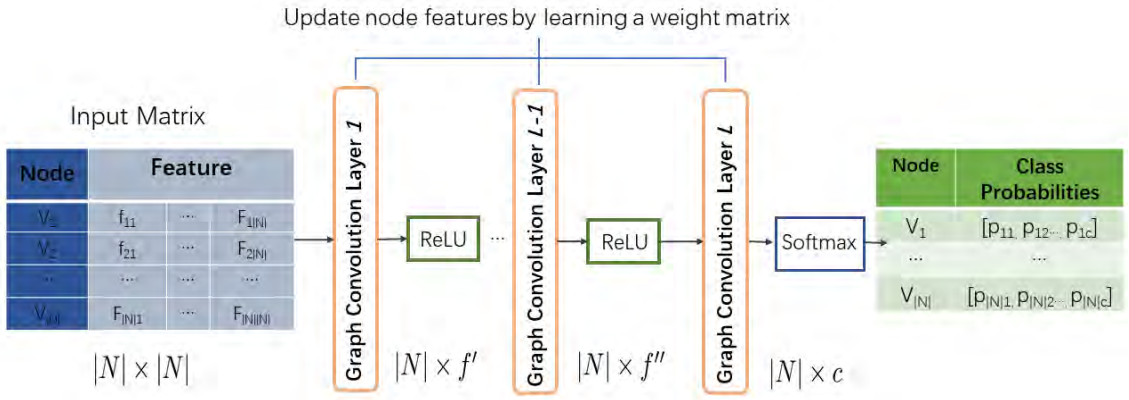


Figure 3.3: The architecture of the Graph Convolutional Network.

Between every two convolutional layers i and $i+1$, we need to learn a weight matrix W^i that is shared by all nodes. Suppose L^i and $L^{(i+1)}$ denote the output of the i-th and $(i+1)$-th layer, respectively. Their relationship is defined as:

$$L^{(i+1)} = f(L^i, A) = \sigma(AL^iW^i), \tag{3.14}$$

where A is the adjacency matrix of the graph; σ is an activation function and we often use *ReLU*.

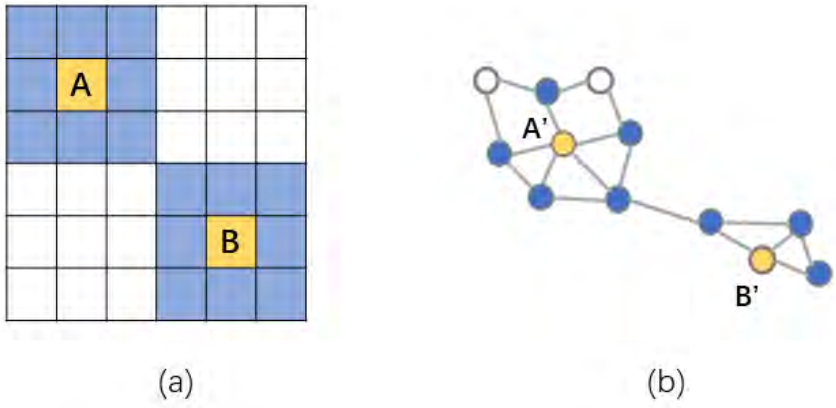


Figure 3.4: Comparing the neighborhood of objects in: (a) low-dimensional regular grids and (b) high-dimensional irregular structures. Neighbors of A, B, A' and B' are highlighted by the blue color.

The reason that A is embodied in Equation 3.14 is because graphs are high-dimensional irregular structures. As shown in Figure 3.4, objects in low-dimensional regular grids, such as images, have a fixed number of neighbors while this is not the case for graph entities. Figure 3.5 provides an example of calculation. The graph consists of three nodes, and we assume each node has a three-dimensional feature vector at layer i, n_{xy} denoting the $y\text{-}th$ feature component of the $x\text{-}th$ node. W^i is in the shape of 3×2, hence the new node feature should be two-dimensional. We can see that without multiplying A, L^iW^i does not update node features by considering the neighboring information. However, the feature of the node itself is ignored in AL^iW^i. Therefore, following manipulations need to be made:

- Add an $|N| \times |N|$ identity matrix I to A to take the node itself into consideration;
- Normalize A to balance the impact of features between high-degree and low-degree

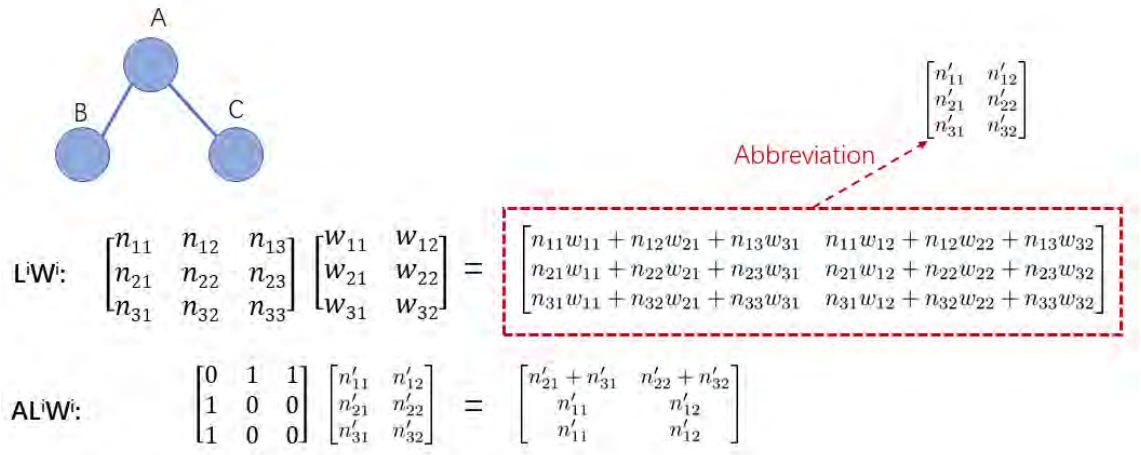


Figure 3.5: An example of calculating the output of a graph convolutional layer.

nodes.

Equation 3.14 should then be transformed into:

$$L^{(i+1)} = f(L^i, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}L^iW^i), \tag{3.15}$$

where $\hat{A} = A + I$ and $\hat{D}$ is the degree matrix of $\hat{A}$.

We use the output of the last convolutional layer as the vector representations of nodes which are also known as embeddings. Each vector is c-dimensional. To get the graph-level representation, we average all node vectors:

$$v_G = \frac{1}{|N|}\sum_{i=1}^{|N|} v_i, \tag{3.16}$$

here v_G and v_i represent the graph and vetex vectors, respectively.

3.3.3 Similarity Measurement

The similarity between graph G_1 and G_2 can be reflected by the distance between their corresponding vectors v_{G_1} and v_{G_2} that are obtained by using either of the foregoing methods. Following distance metrics can be used:

$$\text{Cosine distance} = 1 - \frac{v_{G_1} \cdot v_{G_2}}{||v_{G_1}||||v_{G_2}||}, \tag{3.17}$$

$$\text{Jaccard distance} = 1 - \frac{|v_{G_1} \cap v_{G_2}|}{|v_{G_1} \cup v_{G_2}|}, \tag{3.18}$$

$$\text{Canberra distance} = \sum_{i=1}^{n} \frac{|v_{G_1 i} - v_{G_2 i}|}{|v_{G_1 i}| + |v_{G_2 i}|}. \tag{3.19}$$

The similarity score can be computed as the reciprocal value of the distance. We use the cosine distance in this chapter. $||\cdot||$ in Equation 3.17 denotes the vector length. The distance is 0 when v_{G_1} and v_{G_2} are the same, and 2 when vectors point to opposite directions. The range of the Jaccard distance is $[0, 1]$ and $|\cdot|$ in Equation 3.18 represents the set size. Canberra distance is good at capturing tiny differences between vectors. $|\cdot|$ in Equation 3.19 computes the absolute value and n is the number of components in each vector. Each fraction component of this distance is in the range of $[0, 1]$.

3.4 Visualizing Community Graphs

The goal is to improve the expressiveness of community graphs so that users can achieve a visual understanding of how similar the communities are. Graphs are drawn as node-link diagrams where nodes represent communities. Edges between nodes imply that the corresponding communities contain objects that are originally connected.

Since we regard a dynamic network as a sequence of chronologically ordered static networks, we analyse the requirements of visualization from two perspectives:

- *Spatial dimension*: demonstrate the similarity between communities. How many communities have the similar structures? Is the pair-wise similarity between communities evenly distributed, or is there any community that is very dissimilar from others?

- *Temporal dimension*: track the similarity changes between communities in consecutive snapshots. Is there any identical communities? Are they becoming more similar/dissimilar to others?

For the first requirement, we map the similarity between communities to the distance between node positions on the display. Instead of using the force-directed algorithms, we calculate the graph layout by the MDS (Multi-dimensional Scaling) method so that more closely located nodes indicate that the corresponding communities have more similar structures.

We use MDS as a dimension reduction technique for projecting data items from a high-dimensional space to the Cartesian space, because it is easy to perceive and comprehend the relationship between data items on the 2D display. MDS aims to ensure that the distances between the displayed points reflect the similarities between the input data. The extent to which MDS reaches the goal can be computed by a *stress* function [126]:

$$\text{stress} = \sqrt{\frac{\sum\sum(f(x_{ij}) - d_{ij})^2}{\sum\sum(d_{ij})^2}}, \tag{3.20}$$

where $f(x_{ij})$ and d_{ij} measure the distance between item i and j in the original space and the space after transformation, respectively. In our implementation, we use the Euclidean distance for both of them. The input data are vector representations of graphs. The lower the stress, the more desirable the MDS result.

For the second requirement, we juxtapose snapshots horizontally along the timeline, which is convenient for visual comparison. Each snapshot shows the MDS layout of the community graph. To reveal graph evolution at the community level, we need to do pair-wise comparisons between communities of every two consecutive snapshots, to see if there are any identical communities. Community C_1 and C_2 are deemed as identical if the *ratio* of their node intersection is greater than a threshold,

$$ratio = \min(\frac{|C_1 \cap C_2|}{|C_1|}, \frac{|C_1 \cap C_2|}{|C_2|}), \tag{3.21}$$

where $|\cdot|$ denotes the number of nodes of the community and the operator $\cap$ calculates the common nodes among two communities.

We use color mapping to show the temporal consistency of communities. Namely, nodes corresponding to identical communities are filled with the same color across time steps. Nodes representing different communities are distinguished by different colors.

3.5 Case Studies

We conduct two case studies to evaluate the modules of our framework. In the first case, we focus on comparing the performance of the community detection methods on a static network. In the second case, we present an application of the framework on a real dynamic network.

3.5.1 The Experiment Platform

We carry out experiments on a private cloud platform, and the benefits include:

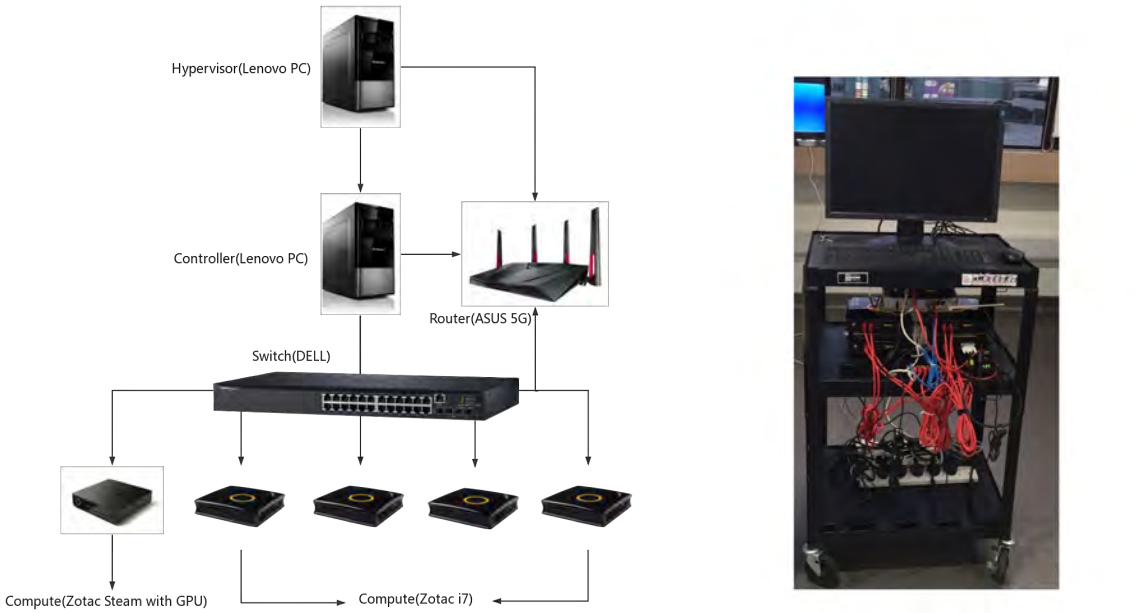
- *Resource pooling*: a cluster of physical machines provide combined capabilities for efficient data processing;
- *Scalability*: the system accommodates to heavier loads by flexibly embodying new servers;
- *Sharing*: various tools are installed and multiple users can access them simultaneously without worrying about the configuration.

3.5.2 The Hardware Platform

The cloud platform is deployed on OpenStack [127] with seven machines, including two PC towers and five Zotac devices. All these machines can access a high bandwidth router through a Dell switch. As shown in Figure 3.6a, the PC with 8 GB of RAM acts as the hypervisor, while the one with 16 GB of RAM plays the role of a controller. we install basic Openstack services on the controller node, such as *Keystone* for identification, *Glance* for image storage and *Horizon* for dashboard access. The controller also manages the schedule of tasks. The cloud compute nodes are four Zotac i7 machines plus one Zotac steam machine with a high-performance GPU. The Zotac i7 machines are individually equipped with 8 cores and 16 GB RAM, and the steam machine has 4 cores and 16 GB RAM. Figure 3.6b presents the real demo.

A virtual machine (VM) has the following resource setting by default: 4 GB RAM, 2 VCPUs, and 5 GB of disk space. VMs are the access points through which users interact with the cloud system. Users can freely create, configure or delete one or multiple VMs. They can also establish VM clusters with diverse topologies.

Regarding softwares, we use python with NetworkX to manipulate and analyse graph data. The deep learning platform we use is Pytorch. Visualizations of community graphs are drawn by JavaScript and D3.



(a) (b)

Figure 3.6: The cloud platform infrastructure

3.5.3 Case 1: Static Network

We downloaded the Amazon product co-purchasing network data [128] from the SNAP website [129]. The network depicts the relationship of which products are frequently co-purchased by customers. The data package also contains the ground-truth communities. Based on the product category provided by Amazon, each connected component in a category is regarded as a ground-truth community.

Data processing. The dataset provides 5000 high-quality communities, but many of them turn out to be duplicates. After removing all duplicates, we have 1517 of them left, and we also find that these are overlapping communities. Since we are only interested

in non-overlapping communities, we assign each node uniquely to the largest one of the communities that it has involved with. This operation causes certain small communities disappear. Finally, we have 1163 communities left as the ground truth.

Our objective of study is to compare the result of the three community detection methods with the benchmark. Therefore, we retrieve the sub-network induced by the nodes of the 1163 ground-truth communities, and apply those methods to this sub-network which contains $16\,716$ nodes and $48\,739$ edges.

Comparative analysis. The statistics about the detection results of the three methods are displayed in Table 3.1. We can see that the Louvain method obtains the most approximate number of communities (i.e., 1121) to the ground truth, while the label propagation method returns the largest number of communities (i.e., 1753). Meanwhile, we find that the former method achieves the highest NMI (i.e., 0.9906) and the latter method has the lowest (i.e., 0.9507). Given that the range of NMI is $[0, 1]$, all these methods have achieved good results as their NMI values exceed 0.9. In terms of the time cost, the Louvain method is the most time-consuming, and the label propagation method almost finishes immediately. The Infomap algorithm achieves moderate performance among all metrics.

Figure 3.7 contains the histograms showing the size distribution of communities. Rugs on the horizontal axis denote the size of individual communities. We set the number of bins to 20. Like the benchmark, most of the communities detected by the algorithms have less than 50 members. In the benchmark, there are two communities whose sizes are larger

than 300. The Louvain algorithm finds a community which has about 230 members, but the Infomap and label propagation method fail to find large communities. Therefore, we believe that the Louvain algorithm achieves the best performance in this case.

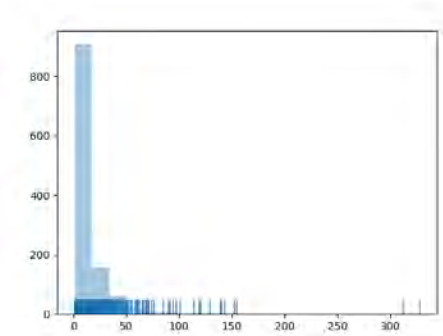
	Louvain	Infomap	Label propagation
$\|C\|$	1121	1424	1753
Time	4.42	0.50	$1.59e^{-5}$
NMI	0.9906	0.9655	0.9507

Table 3.1: The number of detected communities ($|C|$), time cost (second) and NMI based on the benchmark of the three community detection methods.

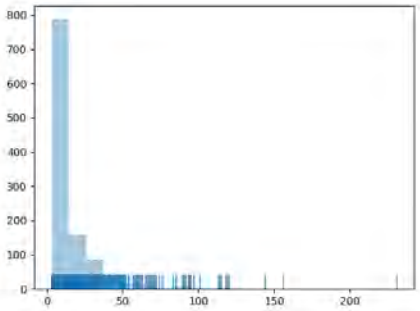
3.5.4 Case 2: Dynamic Network

We retrieve the revision records of Wikipedia during January to December, 2007. Each record is described by a timestamp, an article ID and an editor ID. The time interval is set to one month. Two editors are assumed related if they ever worked on the same article. For each month, editors are ranked by their contribution to different articles. The constructed network at one time step consists of top 1600 editors, and the average number of edges is $41\,609$.

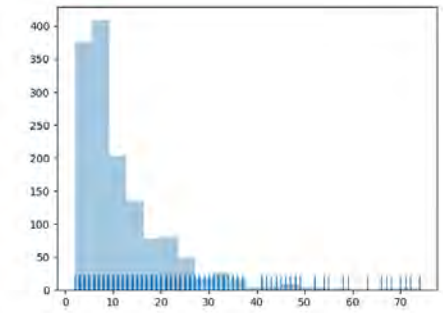
Figure 3.8 displays the original graphs and these drawings are generated by Gephi [130] with a force-directed layout algorithm [131]. Due to the large number of nodes and dense connections, visual overlapping becomes an outstanding problem. Besides, it is impossible to recognize the structural changes by observation.



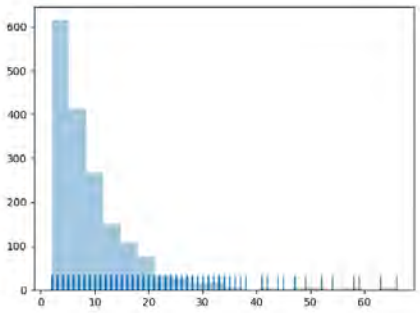
(a) ground-truth



(b) louvain



(c) infomap



(d) propagation

Figure 3.7: The histogram of the distribution of community size.

We adopt the Louvain method for detecting communities at every time step, and the force-directed layout of community graphs are shown in Figure 3.9. Communities are numbered randomly, without suggesting any relationship between communities at different time steps.

Figure. 3.10 shows the matching relationship between communities at consecutive time steps. Corresponding communities can be found in Figure 3.9 through the numbers in the circles. We identify matched communities by applying pair-wise comparisons based on Equation 3.21. The threshold of similarity is set to 0.25. With the increase of time, the

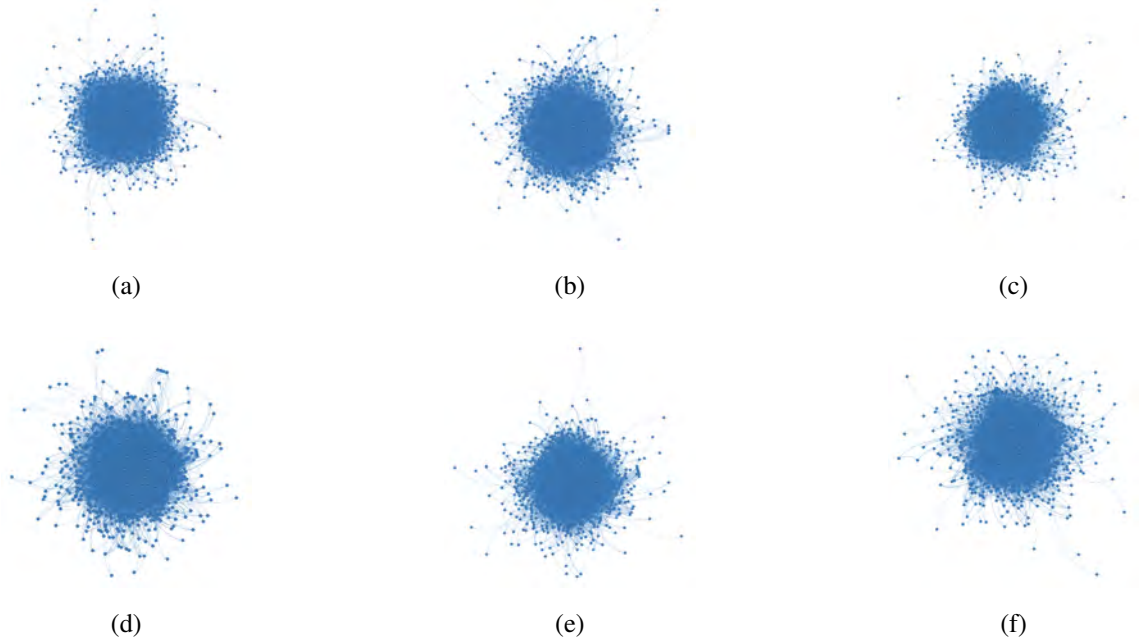
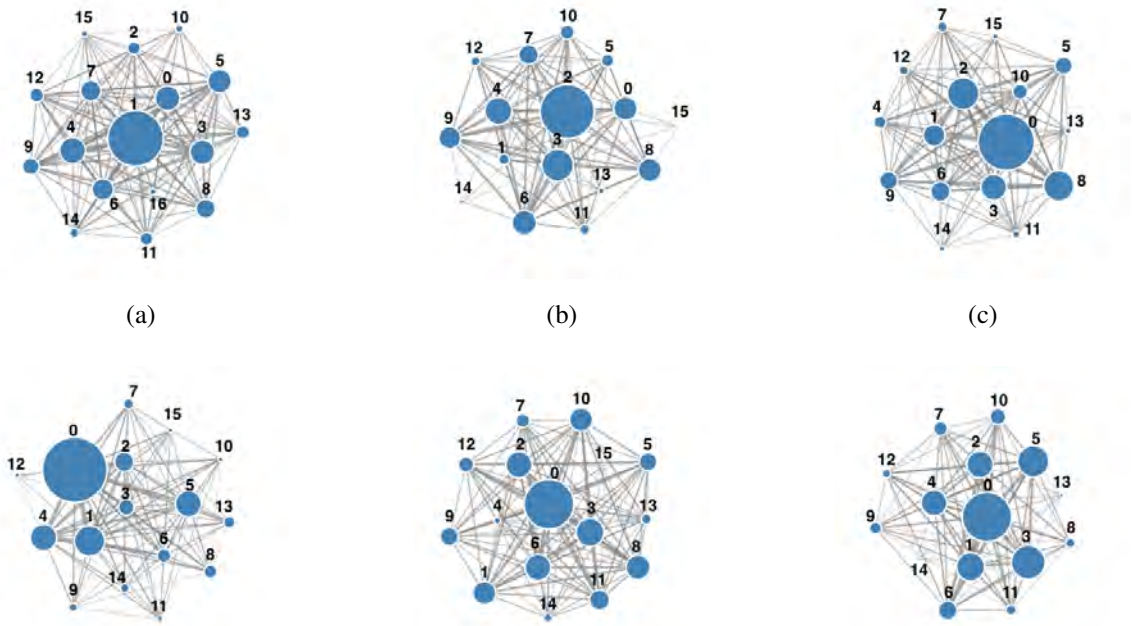


Figure 3.8: Drawings of the original networks at six time steps. Small blue circles and curves denote nodes and connections, respectively.

number of matched communities decreases. There are 8 matching pairs between *step 1* and *2*, but none is found between *step 4* and *5*. *Community 1* is the most endurable and its existence spans four time steps.

The GCN model is then used for obtaining vector representations for the subgraphs induced by communities. We construct a network that consists of two convolutional layers, which means every graph node will learn information from its second-order neighborhood. The number of units in these two layers are 16 and 15, respectively. We train the network repetitively at every time step and the input is the original graph at that time. Graph nodes are labeled by their community indexes. The learning rate is 0.01 and the dropout rate is 0.5. Figure 3.11 shows the accuracy of the model.



(a) (b) (c)

(d) (e) (f)

Figure 3.9: Reduce the visual complexity of original graphs by displaying them at the community level. Circles represent communities.

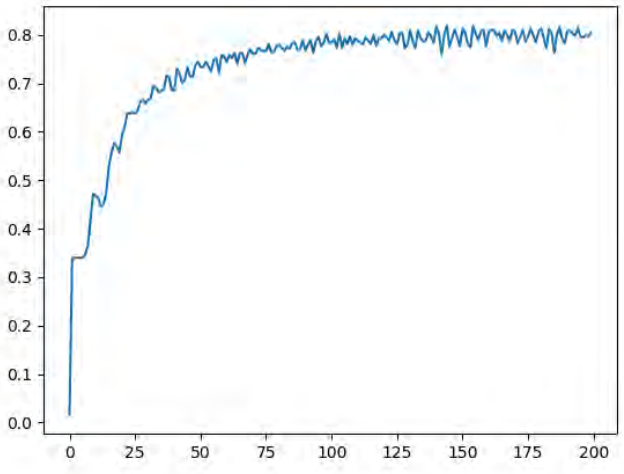


Figure 3.11: The accuracy of the GCN model in 200 epochs.

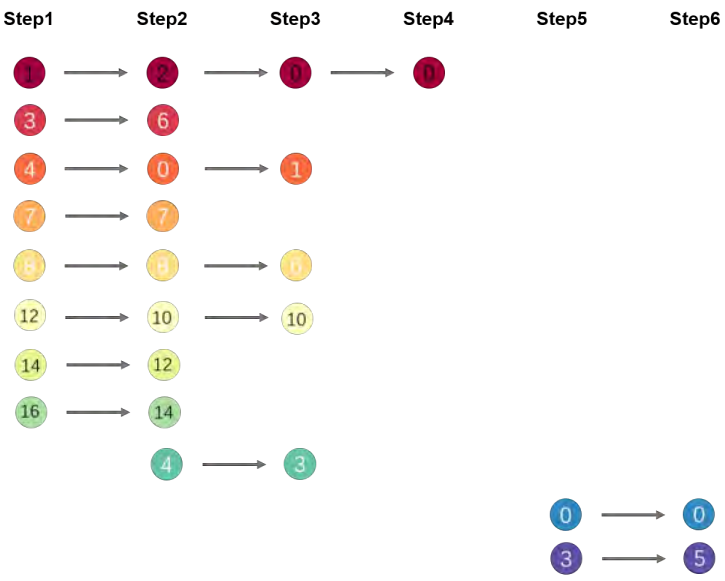


Figure 3.10: The matching relationship between communities at different time steps that are denoted by circles.

After the accomplishment of training, we use the model to get the 15-dimensional node embeddings. To verify if they truely represent the nodes, we project them to 2-dimensional space as shown in Figure 3.12. We can see that nodes belonging to different communities can be well separated.

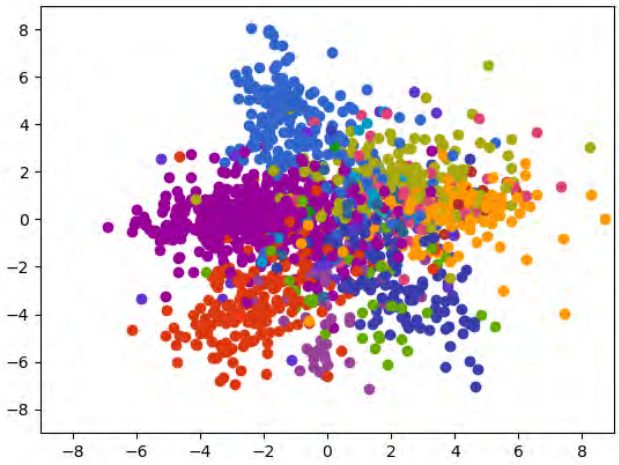
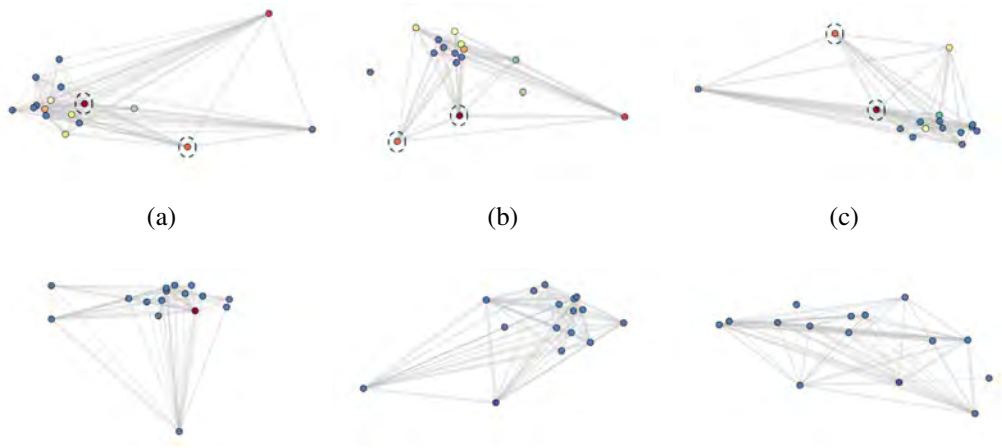


Figure 3.12: The scatter plot of nodes whose coordinates are decided by their two-dimensional embeddings. Different colors are used to differentiate nodes of different communities.

We then compute the community embeddings according to Equation 3.16. The community-level MDS visualization is shown in Figure 3.13. To show the evolution of communities, we fill circles based on the matching relationship demonstrated in Figure 3.10 with the same color scheme [132]. Other circles are filled with the steel blue color by default.



(a) (b) (c)

(d) (e) (f)

Figure 3.13: Visualizing community graphs by MDS. Circles denote communities. The closer the circles, the more similar the corresponding communities.

We find that, from *step 1* to *step 4*, most of the communities are similar, because circles representing them gather together. However, their similarity decreases at *step 5* and *6*, as the distance between circles increases. The two communities marked by dashed lines in Figure 3.13a, 3.13b and 3.13c remain stable and keep having very different structures.

3.6 Conclusion

In this chapter, we present a framework for visualizing the similarities between graph communities. In the three main modules of the framework, we integrate multiple advanced methods for detecting communities and calculating graph similarities. To evaluate the effectiveness of these methods, we conduct two case studies on real data. The similarities between communities are mapped to the distances between nodes in graph drawings

so as to let users obtain a visual understanding about which communities have similar structures. However, in current visualizations, community structures are still hidden from view. Hence, for the future work, we plan to provide visual encodings for such structural information.

CHAPTER 4

VISUAL SIMPLIFICATION BY CLASSIFYING GRAPH STRUCTURES

In last chapter, we discuss how to reveal the structural similarities between communities based on the vectorization of graphs. Though users can identify which communities are more similar by viewing their distances in our visualizations, they still lack information about the individual structures of communities. In this chapter, we solve this problem by a classification-based method introduced in Section 4.3. Besides, we realize the smooth animation of graph evolution by detecting communities at a global level, and the details can be found in Section 4.4.

4.1 Introduction

The visual clutter arises when drawing large graphs in a limited display space. Therefore, it is desirable to cluster entities into groups and represent them by single nodes so that we can achieve visual simplicity. Given these simplified node-link diagrams, users can no longer access the topologies of clusters because they have been concealed. We aim to solve this problem by classifying the structures of clusters into user-defined topological patterns. Then, we embed glyphs denoting those patterns into the simplified visualization.

This approach makes it convenient to understand the local structures, search for specific topologies, and capture the structural differences.

Community detection algorithms can be used to cluster densely connected vertices. Extensive researches have been conducted on static graphs [68, 67]. According to Rossetti et al. [2], dynamic community detection techniques are categorized into three main types: (1) Instant Optimal [133], (2) Temporal Trade-off [88] and (3) Cross Time [134]. From (1) to (3), the optimality of the community structures decreases while the temporal stability increases. We propose a cross-time algorithm based on the concept of the SuperGraph [40]. The communities detected from the SuperGraph are used as references for retrieving communities at individual time steps. Using this approach, we can achieve high temporal consistency at a low time cost.

We provide topological patterns to approximate the original community structures, and they imply the backbone shapes of graphs. Li et. al defined several topological patterns [135]. Similar to their work, we adopt four patterns: *chain*, *loop*, *clique*, and *ego-centric*, based on the prevalent topologies in graph theory.

To classify graph structures into topological patterns, we take graph layouts rather than the original graphs as the input and train a multi-class model for prediction. Because we find that a determined layout algorithm generates similar drawings for graphs of similar structures, as shown in Figure 4.1. The benefit of this approach is that we can eliminate the exhaustive calculations involved in traditional methods. Notably, the layout algorithm and its configurations must remain constant regardless of the training or prediction stage.

Graph drawings vary considerably when using different layout algorithms or parameters, which might reduce the classification accuracy.

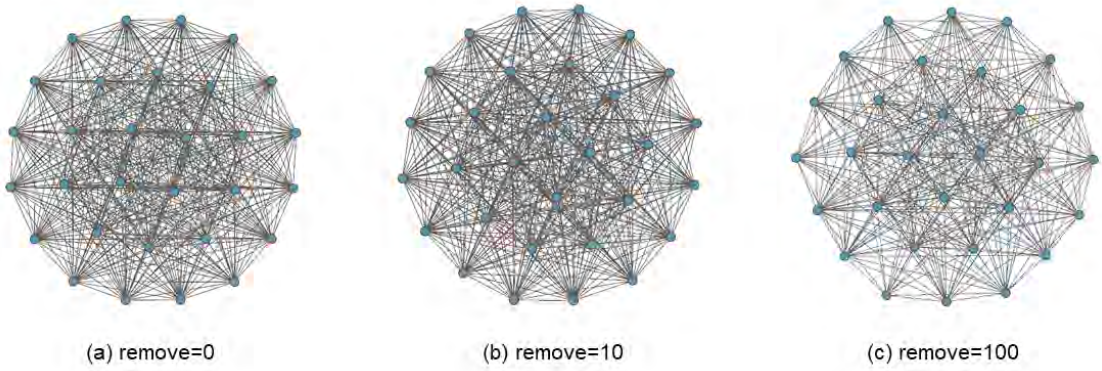


Figure 4.1: Using the same parameters, the Fruchterman-Reingold layout algorithm generates similar layouts for graphs of similar structures. From (a) to (c), we create the graph data by removing a certain number of edges from a complete graph on 30 vertices.

We design compact glyphs to represent the topological patterns and integrate them into the node-link diagrams. Node positions are computed at a global level so as to achieve the visual smoothness. We use animations to display the evolution of time-varying graphs. Since animations do not provide good support for comparing different snapshots, we place the previous and current topological patterns concentrically in one animation frame. The results of the case studies show that our method produces animations that result in minimum visual discontinuities. Meanwhile, users can identify both the global and local structural changes. Specifically, our contributions are as follows:

- We map graph communities to topological patterns, which is an efficient way to navigate users through diverse graph structures. In this work, we define four pat-

terns. However, new patterns can be easily added by re-training the classification model;

- We propose a layout-based classification method and train a model by learning the visual characteristics of the layout images. The model can then be used to predict the topological patterns of arbitrary graphs;

- We improve the visual stability by determining the layout of graph drawings at a global level. Thus, no abrupt changes occur between consecutive snapshots. Pentagonal glyphs represent the topological patterns, and their inter-transformation procedure keeps steady.

- We implement the exploration of graph structures based on simplified node-link diagrams. It is convenient to locate communities that have similar structures and make comparisons between different snapshots.

4.2 System Overview

We build a visual system that provides simplified representations of large and complex time-varying graphs. This system aims to assist users in tracking the evolution of graph structures and in searching for topologies of interest. We reduce the complexity of visualization and analysis by focusing on graph communities. Figure. 4.2 shows the main stages of our work.

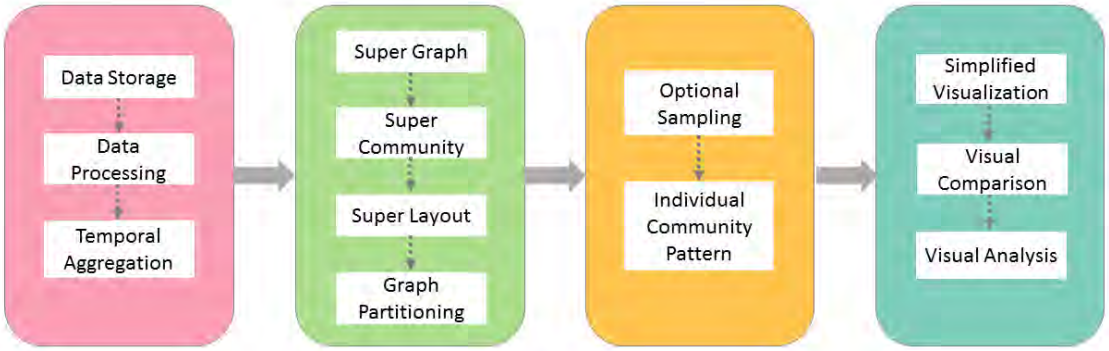


Figure 4.2: The pipeline of implementing the visual system. It consists of four stages: data processing, graph partitioning, structure classification and visual simplification.

In the data processing procedure, we apply temporal aggregation to obtain the desired time granularity (e.g., a year).

In the graph partitioning step, we construct a SuperGraph (SG) by combining the snapshots at all timestamps. The Super Community (SC) is a set of communities detected from the SG. Taking the communities in SC as nodes, the corresponding layout determined by a force-directed algorithm [136] is called the Super Layout (SL). At individual timestamps, we compute the communities and their positions on the display by referencing the SC and SL, respectively. SG, SC and SL are the key to producing smooth animations. The details of this process are explained in Section 4.4.1.

It is worth noting that calculating SG, SC and SL results in a smooth animation. We do not apply community detection and force-directed layout repeatedly to each snapshot. The algorithms need to be executed only once at the global level. To detect communities, we adopt the Louvain algorithm [67], which is one of the fastest implementations, with a

complexity of $O(n \log n)$, where n is the number of vertices.

The classification of structures depends on layout images. However, graph drawings of large communities are likely to suffer from visual overlaps, and machines may fail to distinguish them. We solve this problem by adding an optional sampling step for communities whose sizes exceed the limit.

In Section 4.5, we realize the simplification of visualizations by regarding communities as the minimum visual unit. However the local information is lost. Therefore, we enrich the node representations with glyphs that denote the topological patterns. Consequently, users can still compare different communities and capture their structural changes.

4.3 Structural Classification

4.3.1 Pattern Description

A topological pattern represents a specific type of graphs from which we can easily perceive the connectivity features. Suppose n is the number of graph vertices, we define four types of patterns:

- ***chain***: n vertices are linearly connected by $(n-1)$ edges;
- ***loop***: a n-path of n vertices with two end-vertices connected;
- ***clique***: n vertices are fully connected;

- ***egocentric***: a vertex is located in the center and all the remaining vertices connect to it.

In the *chain* pattern, vertices are connected in a sequence, and they potentially reflect the ordering information. The *loop* pattern contains a closed path, and the connectivity is sparse. Conversely, vertices are densely connected in the *clique* pattern. The *egocentric* pattern has been extensively studied in the context of social networks [107, 109]. It leads us to find dominant entities.

We define the patterns according to the graph structures that researchers are usually interested in [137]. Meanwhile, they are very different in topology, making their layouts easy to distinguish. The shapes of the patterns are illustrated in the left column of Figure 4.3, using five or six nodes for simplicity. Alternative patterns can be the ones shown in Figure 4.4.

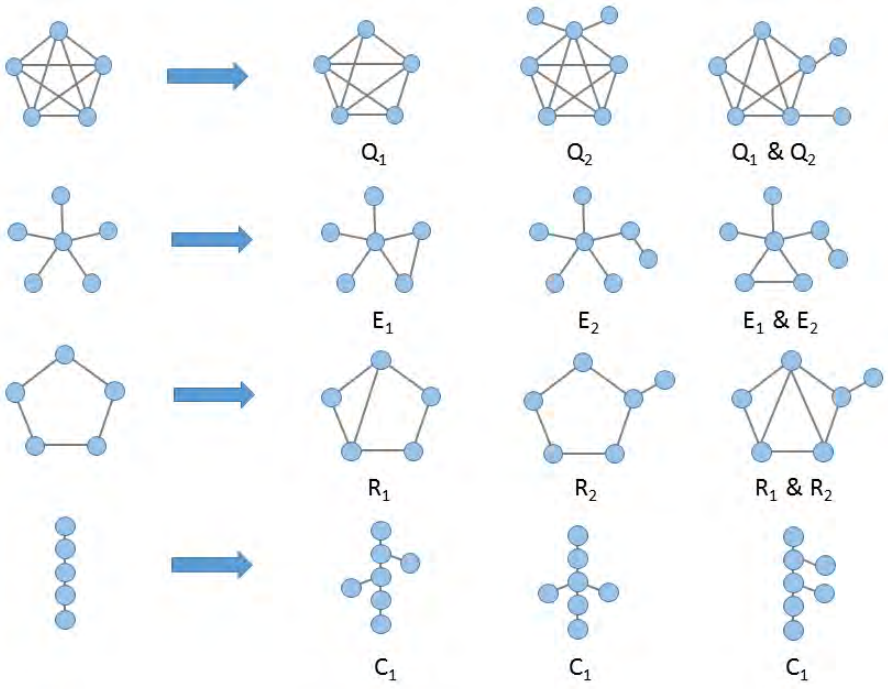


Figure 4.3: Pattern drawings (left column) and variants of the *clique* (first row), (b) *ego-centric* (second row), (c) *loop* (third row) and (d) *chain* (last row) pattern.

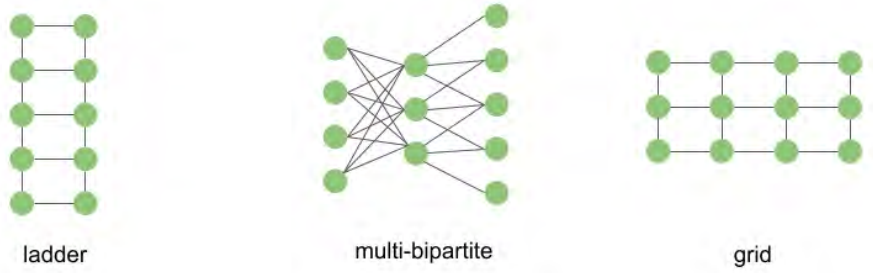


Figure 4.4: Alternative topological patterns.

4.3.2 Sampling

We limit the size of communities by carrying out random sampling strategies on vertices, edges, and explorations [62]. There are two reasons: first, large graphs cause the visual clutter, and the layout-based classification might fail; second, users can hardly verify the classification result by observing the complex graph drawing.

Following sampling methods are available. Users can compare their results and select one of them:

Random Vertex Sampling: randomly select a set of vertices from the graph and return the subgraph induced by these vertices;

Induced Random Edge Sampling: randomly select a set of edges from the graph and return the induced subgraph on vertices incident to at least one of the selected edges;

Random Walk Sampling: randomly select a starting vertex and then simulate a random walk. The sampling result includes all visited vertices and edges.

Sampling results are evaluated by comparing their structures with that of the original graphs. A higher structural similarity suggests a better result. The similarity score can be calculated by using the method proposed in Netsimile [123]. We have introduced the implementation of Netsimile in Section 3.3.1 of Chapter 3. It converts a graph into a signature vector that encodes the structural characteristics of the graph.

The similarity score equals to the reciprocal of the *Canberra distance* between two

graph signature vectors V and V':

$$\text{Canberra distance}(V, V') = \sum_{i=1}^{n} \frac{|V_i - V_i'|}{|V_i| + |V_i'|}, \tag{4.1}$$

where n denotes the number of components in vectors, and $|\cdot|$ calculates the absolute value. The run time complexity of Netsimile is linear on the number of edges, and the score value lies in [0,1], where 0 suggests totally different structures, and 1 means that the graph structures are the same.

4.3.3 Classification Model

We classify graphs induced by all communities into four types as we defined in Section 4.3.1. Graphs that belong to the same class may vary a little in the structure, but they all have similar skeletons.

Traditional solutions to graph classification need to transform graphs to numeric representations that facilitate the execution of machine learning tasks [121, 123]. However, they involve calculations of many topological properties and require for iterative graph searches. Hence, these methods are time-consuming and not applicable to large graphs. Kwon et al. [121] pointed out that, given a determinate layout method, topological similarities contribute to perceptual similarities. Hence, the classification result of layouts should be consistent with the classification result of the corresponding structures.

Our method predicts the class label of a community by passing its layout image to the classification model. During the implementation of our system, the force-directed

algorithm [136] and the involved parameters such as attractive and repulsive forces remain the same.

Model: the multi-class model is built on top of a pre-trained neural network, the ResNet-18 [138]. It optimizes a residual mapping instead of an identity mapping. Apart from the basic building blocks, there are bottleneck modules, reducing the dimensions of deeper networks. ResNet-18 is a stack of basic building blocks. Cross-entropy is used to calculate the loss. The learning rate is 0.001 for the stochastic gradient descent optimizer. We add a softmax layer to normalize the output to a probability distribution over the four classes and choose the class with the highest probability to label the input sample.

Training Set: Initially, we build the training set with graphs that have the rigorously same structures with the topological patterns. We call these graphs as *templates*. The number of vertices ranges from 5 to 100. So, there are 384 (i.e., $(100 - 5 + 1) \times 4$) templates in total. To expand the training set, we use these templates as seeds, and create their variants by running the following operations:

- *clique*: randomly remove existing edges (*Q1*) or add periphery vertices to templates (*Q2*);
- *egocentric*: randomly connect existing alter vertices (*E1*) or add periphery vertices to templates (*E2*);
- *loop*: randomly add edges between existing vertices (*R1*) or add periphery vertices to templates (*R2*);

- *chain*: add periphery vertices and connect them to non-endpoint vertices (*C1*).

The foregoing operations are executed multiple times until we get 2000 samples for each pattern. Examples of variants are given in Figure 4.3. We obtain graph layouts by using the D3 library [139] and train the classification model on the Pytorch platform [140]. Figure 4.5 shows the accuracy of the model.

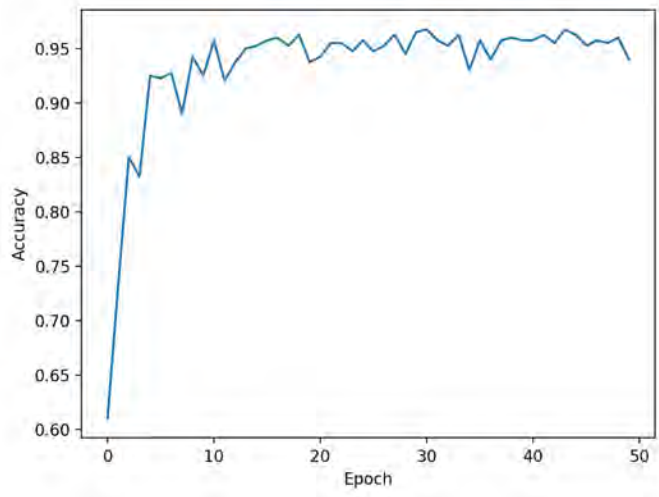


Figure 4.5: The accuracy of the classification model in 50 epochs.

4.4 Smooth Animations

We measure the smoothness of animations by calculating the visual differences between every two consecutive frames. Frequently used metrics include the Mean Squared Error (MSE) and the Structural Similarity Index Metric (SSIM). In this section, we implement smooth animations by: first, preserving the consistency of graph layouts; second, optimiz-

ing the visual design of pattern glyphs.

We give mathematical definitions of terminologies that will be used later.

Definition 4.4.1. ***Time-Varying Graph:*** $G_T = \{G_1, G_2, \ldots, G_t\}$, where t is the number of time steps. For $1 \leq i \leq t$, $G_i = (V_i, E_i)$, representing the snapshot at time i. V_i and E_i are the set of vertices and edges, respectively.

Definition 4.4.2. ***SuperGraph:*** $SG = (V_S, E_S)$, where $V_S = V_1 \cup V_2 \cup \cdots \cup V_t$ and $E_S = E_1 \cup E_2 \cup \cdots \cup E_t$.

Definition 4.4.3. ***Super Community:*** $SC = \{C_1, C_2, \ldots, C_k\}$ is the set of communities detected from SG, where k is the number of communities.

Definition 4.4.4. ***Super Layout:*** SL is the force-directed layout of $C_1, C_2, \ldots, C_k$, consisting of the screen positions of nodes that represent super communities.

4.4.1 Stable Layout

SC and SL are the references for obtaining the communities and the layout of $G_1, G_2, \ldots, G_t$. As shown in Figure 4.6, we firstly compare the set of vertices in SG and G_i, and they are denoted by V_s and V_i respectively. For each vertex in V_i, we then extract the corresponding communities and their positions from SC and SL. Hence, G_i has two communities, C_1' and C_2'. Also, $|C_1'| = |C_1|$, $|C_2'| < |C_2|$ and $|\cdot|$ denotes the number of inner vertices. Despite the size difference between C_2' and C_2, they are at the same position on the display. Hence, our method can produce animations that seldom suf-

fer from abrupt visual changes, except that adjacent snapshots have substantially different membership of communities.

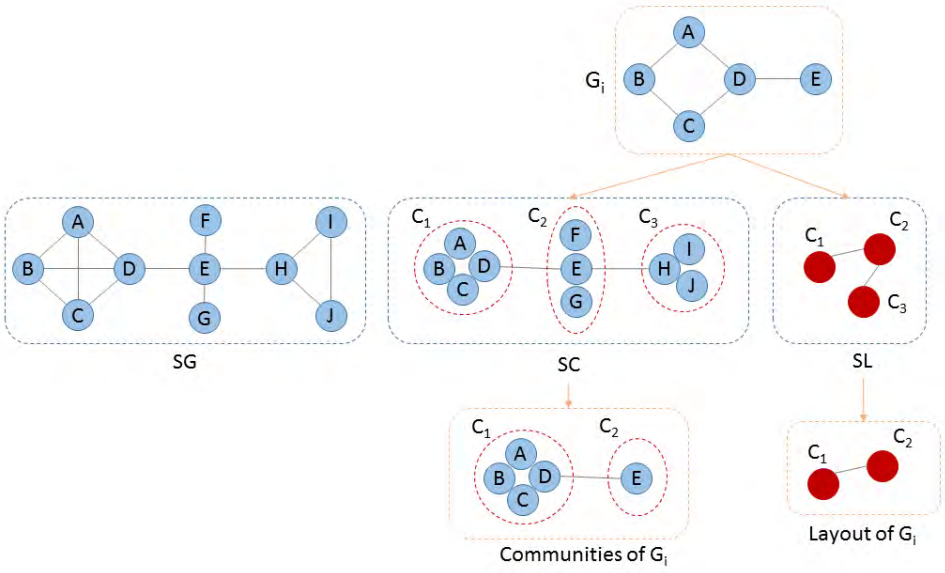


Figure 4.6: For individual snapshots, we extract communities and their positions from SC and SL.

The community detection and the layout algorithms only run on SG and SC. For each snapshot, the calculation time is linear to the number of vertices. Hence, the total run time complexity is $O(|V_S|\, log(|V_S|) + |V_S|)$.

4.4.2 Pattern Transformation

At each time step, we use the classification model to assign topological patterns to communities. Consequently, diverse structures are reduced to four comparable patterns, each represented by a glyph. Users can observe the distribution of the patterns from the sim-

plified visualizations. Besides, it becomes easier to identify communities of similar structures and monitor the evolution of the whole graph.

We use five nodes as the base of a glyph drawing to achieve compactness and aesthetics. Provided that three nodes were used, it would be impossible to distinguish between *loop* and *clique*. Also, it is not impressive to demonstrate the dense connectivity of *clique* by four nodes. Using more than five nodes might increase the visual clutter and decrease the readability.

We improve the visual smoothness by controlling the transformation of glyphs. Glyphs are attached to the circles that denote communities. When the community pattern changes, we do not want to get sharp visual differences. To achieve this goal, we bend the drawing of the *chain* pattern so as to make the four types of glyphs fit a pentagonal form. During the transformation, five peripheral nodes remain static and only edges change.

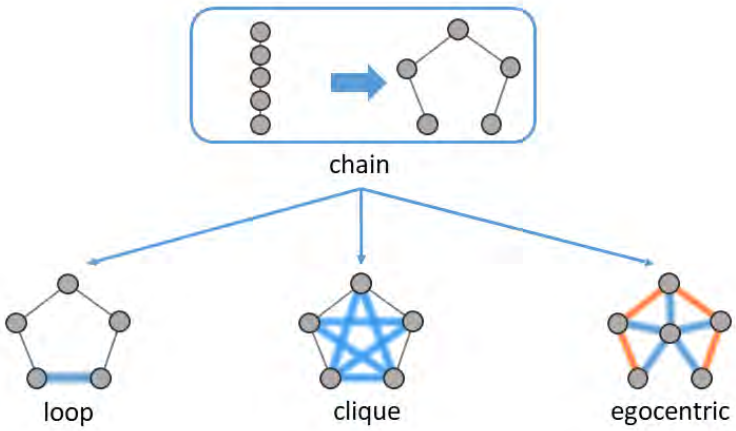


Figure 4.7: The *chain* pattern smoothly transforms to other patterns. Emerging edges are marked by *blue* halos and *orange* halos represent disappearing edges.

As shown in Figure. 4.7, if the *chain* pattern transforms to a *loop* pattern, only the bottom edge needs to be added. However, if the target pattern is *egocentric*, several edges need to be added and removed simultaneously. The transformations between other pairs of patterns are implemented similarly as shown in Figure 4.8. From the *egocentric* pattern to other patterns, the central node should also be removed. The pentagonal appearance of pattern glyphs allows us to have a fluid visual perception when watching the animation.

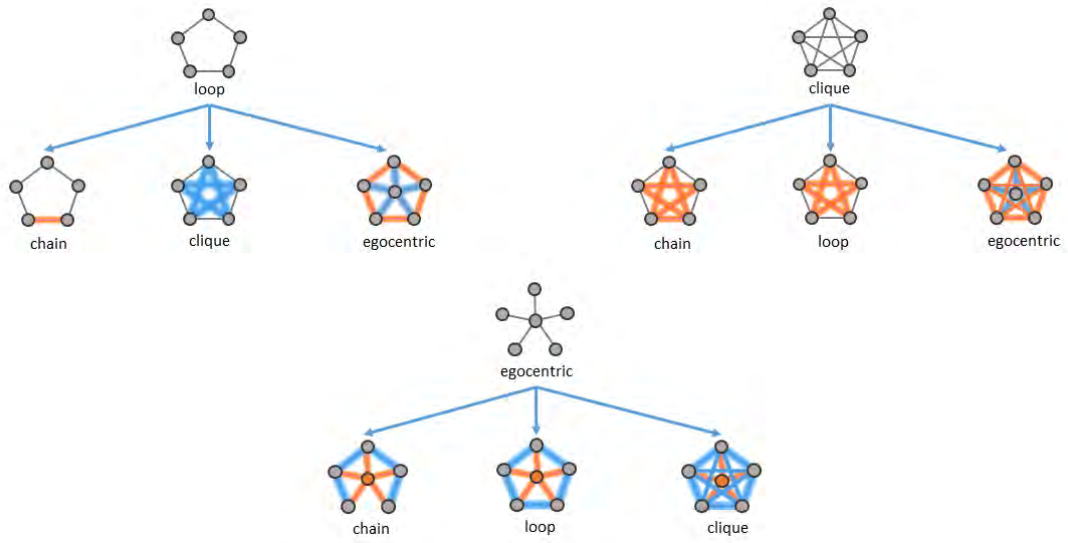


Figure 4.8: The manipulation of glyphs shows the transformations from the *loop*, *clique* and *egocentric* pattern to other patterns.

4.5 Visualization

Our visual analytic system aims at assisting users to perform the following visual tasks that are related to exploring the structural changes of graphs:

- **Graph composition (T1)**: discover the graph evolution at a high level. How many communities does the graph have at each time step? Are they densely connected or not?

- **Community importance (T2)**: measure the importance of a community by counting the number of vertices belonging to it and the total weight of its edges connected to other communities.

- **Community duration (T3)**: find stable and dynamic communities. More stable communities stay longer during the whole period?

- **Community evolution (T4)**: discover the graph evolution at a local level. How does the size and structure of a community change temporally? What is the topological pattern that a community matches most of the time? At what time points does a pattern transformation occur?

- **Topological distribution (T5)**: find communities with similar/different structures. What is the distribution of topological patterns in one snapshot? How does the distribution evolve temporally? Are the patterns always evenly distributed, or some of them dominate?

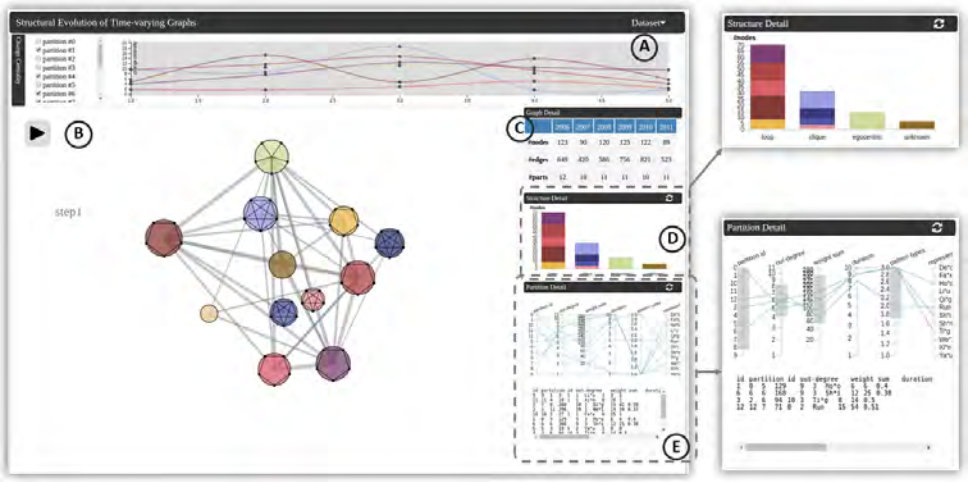


Figure 4.9: The interface of the visual analytic system. The data in use is from the DBLP dataset. (A) represents the change centrality of communities, which quantitatively measures the structural changes between consecutive snapshots. (B) plays the smooth animation of time-varying graphs. (C) presents the temporal information of graph statistics. (D) shows the distribution of topological patterns at a specific time. (E) reveals the community details with multiple metrics.

Figure. 4.9 shows the interface of our system that is implemented by using JavaScript and D3.js [139]. The animation is played in (B). Rich statistic information of the graph and communities is provided to support visual analysis. Tables and charts in (D) and (E) can be updated when users select a certain time step. Specifically, users can access the distribution of the four topological patterns and the membership of communities in (D). In (E), multiple metrics are used to describe the characteristics of communities including the duration of communities' existence, the type of the topological patterns, the total weight of the inner connections, etc. As Parallel Coordinates (PC) are effective visualization tools for analyzing multivariate data, we use them to present these characteristics. PC consist

of multiple parallel axes, each denoting a characteristic. A community is represented by a polygonal line that connects the vertices on axes. The vertices correspond to the characteristic values of the community. By inspecting the intersection of the polygonal lines and the individual axes, users can easily understand the value distribution of each characteristic. Besides, users can perform queries on different ranges of one or multiple characteristics by brushing on the axes. The explanation of the change centrality in (A) can be found in Section 4.5.1.

4.5.1 Variation Trend

We can locate fundamental changes of community structures by capturing the visual transformation of topological patterns. But it is difficult to perceive the extent of variation by observation. Besides, minor changes may not trigger the transformation. Therefore, we provide view (A) in Figure 4.9 to show the quantitative measurement of variation.

The metric we use is *Change centrality* [141]. It measures the topological changes and the temporal changes of a vertex i simultaneously. From time t_1 to t_2, the calculations are done by:

$$r^n_{t_1,t_2}(i) = \frac{\left|N^n_{t_1}(i) \Delta N^n_{t_2}(i)\right|}{\left|N^n_{t_1}(i) \cup N^n_{t_2}(i)\right|}, \tag{4.2}$$

$$cc_{t_1,t_2}(i) = \frac{1}{2} \sum_{n=0}^{e_i} \frac{1}{2^{(n+1)}} r^n_{t_1,t_2}(i), \tag{4.3}$$

where e_i is the maximum graph distance from i to any other vertices; $r^n_{t_1,t_2}$ is the *change*

ratio defined in Equation 4.2; $N_{t_1}^n(i)$ is the set of neighbours that are n steps away from i at time t_1; $|\cdot \Delta \cdot|$ calculates the number of nodes that are added to or removed from the n-step neighbors of vertex i; $|\cdot \cup \cdot|$ calculates the number of nodes in the union of two sets. The change centrality $cc_{t_1,t_2}(i)$ of a vertex i is defined as a weighted summation of the changes of the adjacent neighbours, the adjacent neighbours of the latter and so on. The weight is defined as:

$$\text{weight} = \frac{1}{2^{(n+1)}}. \tag{4.4}$$

The weight decreases quickly with the increase of n, hence we make farther neighbours play a less important role in the metric by multiplying the weight. Besides, with the weight definition in Equation 4.5, the change centrality converges with its value in the range of $[0, 1]$. We then define the change centrality of a community as the summation of the changes of all its members:

$$cc_{t_1,t_2}(community) = \sum_{i=1}^{|C|} cc_{t_1,t_2}(i), \tag{4.5}$$

where $|C|$ is the number of members.

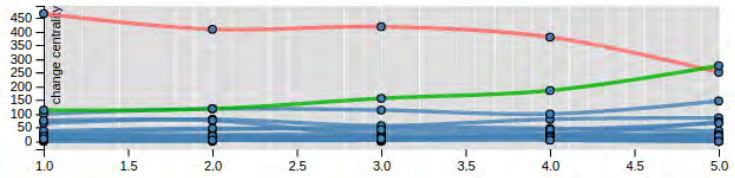


Figure 4.10: Circles mark the change centrality values of communities. A curved line shows the changing trend of a community's change centrality. The data in use is from the Wikipedia dataset.

In Figure 4.10, the *x-axis* and the *y-axis* denote *time* and *change centrality* respectively.The centrality value at $x = t$ represent the extent of changes from t to $t + 1$. The larger the centrality value, the more changes occur to the entities of a community. We use curved lines to connect circles denoting the same community so that it would be convenient to observe and compare the changing trend. For example, the community represented by the red line in Figure 4.10 experiences much more changes than others. But, it tends to be stable because the centrality value decreases. The green line implies that the corresponding community is becoming more dynamic.

4.5.2 Community Evolution

As frames are displayed one by one in animations, it is difficult to compare the differences between them. To alleviate this problem, we place two topological patterns of a community concentrically in a circle, one is for the previous time step, and the other is for the current step. As a result, the visualization maintains compactness, and users can directly perform visual comparisons. We assume that the current patterns should attract

more attention. So they are placed in outer circles and occupy larger space. The previous pattern is half the size of the current one. As shown in Figure 4.11, previous patterns locate in inner circles. It is worth noting that though *clique* and *egocentric* patterns are partly covered by the inner area, we can still recognize them.

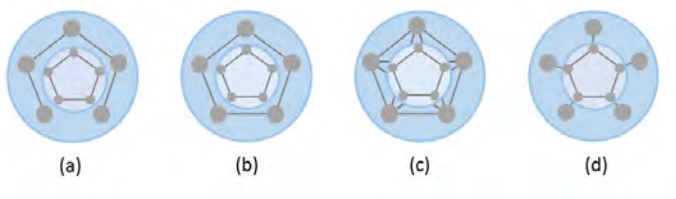


Figure 4.11: Pattern comparisons of two consecutive time steps. Current patterns are placed in outer circles, and previous ones are in inner circles. In this example, current patterns are (a) *chain*, (b) *loop*, (c) *clique* and (d) *egocentric*, and all previous patterns are *loop*.

Changes might happen to multiple communities at the same time. For example, new communities appear, and old ones disappear. Even if the glyph remains at the same screen position across continuous time steps, the corresponding community may still involve expansion, contraction, and pattern transformations. To highlight different types of changes, we map them to the visual encodings listed in Table 4.1.

Table 4.1: Visual encodings of the community changes. Emerging communities have no previous patterns. We remove vanishing communities from the visualization. The size increase and decrease are illustrated by solid and dashed circle borders, respectively. The white filling of inner circles indicates the pattern transformation.

Existence	appear	
	disappear	remove from display
Size	increase	
	decrease	
Structure	unchanged	
	changed	

Generally, there are three categories of changes, in terms of the community's existence, size, and structure. Different types of changes can happen to a community at the same time. If a community newly appears, it does not have previous patterns, and we regard that its size is increased. Hence, we remove the inner circle of the glyph and adopt a thick solid border. Conversely, if a community disappears, we wipe it out from the display. The size of a community equals to the number of the inside vertices. If the size is unchanged or increased, we use solid circle borders. Otherwise, we use dashed borders instead. The *white* color is reserved for filling the inner circles of glyphs which denote communities that undertake pattern transformations. If the topological pattern of a community is unchanged, then the outer and inner circles of the glyph are filled with the same color. In this work, we use different types of lines and colors to distinguish various

changes. As the background color of circles might make the lines difficult to read, we can decrease the opacity of colors to increase the contrast. Though alternative shapes for nodes in node-link diagrams can be triangles, squares and so on, we choose circles because we can draw them as circumcircles of pentagons so as to improve the aesthetics and compactness of visualizations.

4.6 Case Studies

To evaluate the performance of our method, we conduct case studies on two datasets. For the Wikipedia Edit History dataset [142, 143], we focus on measuring the time cost of classifying community structures and the visual differences between animation frames. For the DBLP dataset [128], we discover significant findings from the visualizations.

4.6.1 Wikipedia Edit History Data

We retrieved the complete Wikipedia revisions data from January to December 2007, and the time interval was a month. We focused on the relationship between different editors. Two editors were connected if they had ever worked on the same article. Specifically, for each snapshot, top k editors were abstracted as vertices of the network. In this case, k equals to 1600. We ranked editors by the number of different revisions that they had made.

The first row of Figure 4.12 contains node-link diagrams of the constructed networks. They were generated by using Gephi [130] with the same force-directed layout algo-

rithm [131]. We can see that severe visual clutter problems prevent users from identifying network changes.

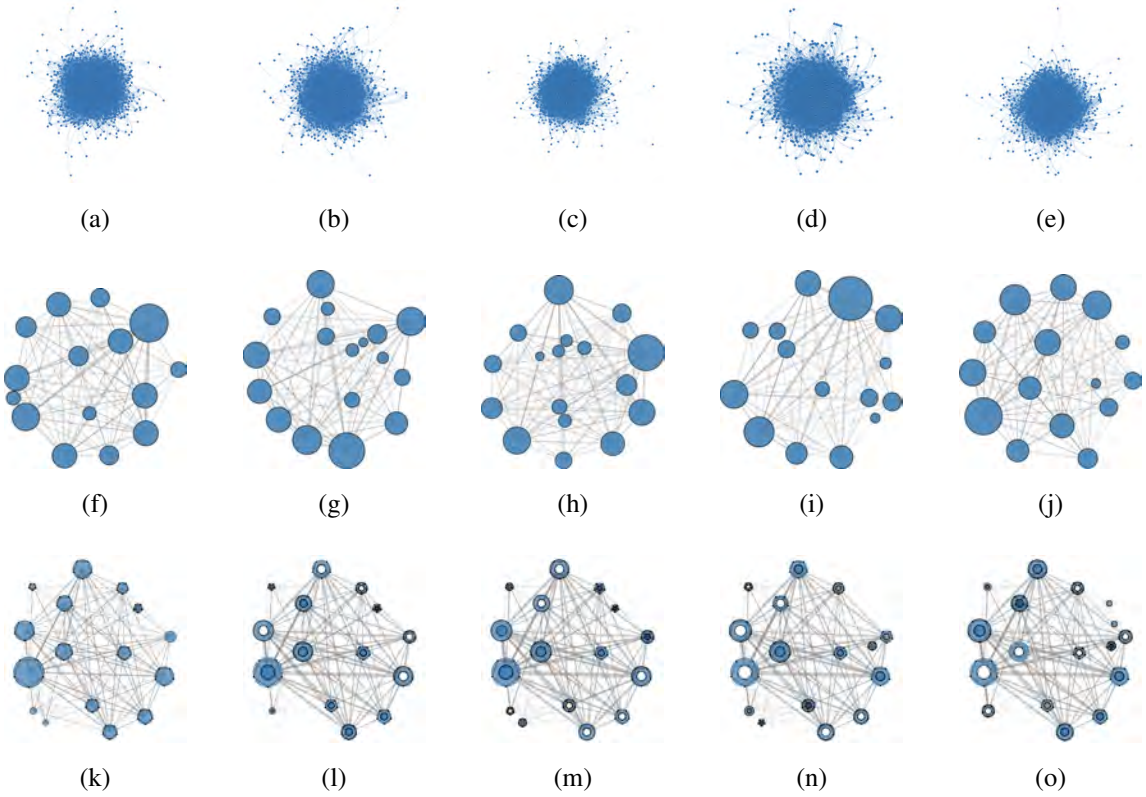


Figure 4.12: Graph drawings of the Wikipedia data in five months, from January to May. The first row shows the original graphs. Nodes and links denote editors and their co-authorship, respectively. For the second and third row, nodes represent communities. Communities and their positions in the second row are computed independently for each time step. The third row presents the visualization results generated by our method.

The third row of Figure 4.12 shows the result of our method. We convert the representations of complex networks to much simpler forms, by clustering closely related vertices into communities and substituting them by a single node. Since the structural information

is hidden from view, we attach pattern glyphs for compensation. Table 4.2 lists the calculation time of community detection, layout, and classification. The classification time is a summation of the time taken to classify all communities at each time step. According to Figure 4.12, there are about 17 communities in each snapshot. The time cost of training the classification model is about 44 minutes and 43 seconds.

Table 4.2: The time cost (in seconds) of community detection, layout and classification.

	detection	***layout***	***classification***
Jan.	0.12	0.004	1.23
Feb.	0.13	0.007	1.87
Mar.	0.18	0.009	1.38
Apr.	0.20	0.007	1.15

We set the size limit of communities to be 100. Users are allowed to select a sampling technique: RV, RE, or RW to reduce the community size. The performance of different sampling methods vary on graphs of different structures. We measure the structural similarity between sampling results and the original graph by Netsimile [123]. Users are suggested to select the sampling method that achieves the highest similarity score. We take one community as an example. It contains about 150 vertices. Figure 4.13 shows its sampling results by RV, RE, and RW in February and the similarity scores from February to May. We can see that for this community, RV achieves the best result.

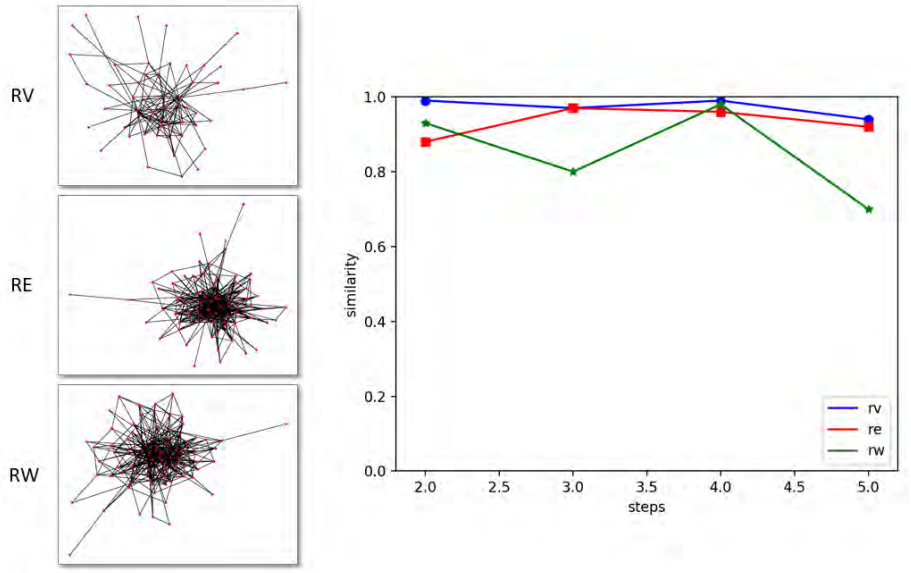
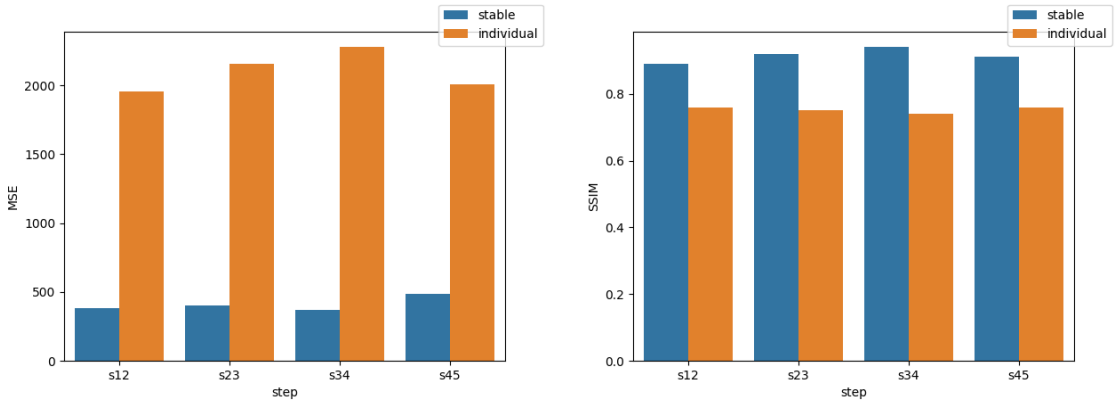


Figure 4.13: Sampling results of a graph by RV, RE and RW. Their structural similarities to the original graph is measured by Netsimile.

To measure the visual smoothness of the animation, we compared every two consecutive frame images by MSE and SSIM. When MSE equals to 0, it implies that two frames are the same. Higher MSE values indicate bigger dissimilarities between frames. The SSIM value lies in $[-1, 1]$, and it reaches to 1 when two frames are the same. Figure 4.14 shows the measurement of visual similarity. $s12$ indicate the value between January and February, and $s23$, $s34$, and $s45$ are likewise. We compared our smooth animation with the animation where community detection and graph layout were conducted separately for individual frames. It is clear that our method produces less visual artifacts.



(a) MSE measures the perceived changes.

(b) SSIM reflects the structural changes.

Figure 4.14: Measuring the visual smoothness of animations with (*blue*) and without (*orange*) calculating SG, SC and SL.

4.6.2 DBLP Data

The DBLP dataset contains the co-authorship between researchers in computer science. We selected an active researcher (i.e., the *ego*) and built an egocentric network by retrieving all other researchers (i.e., the *alters*) who had ever cooperated with him from the year 2006 to 2015. The connections between alters were also considered. There were about 515 authors and 3268 co-authorships involved in 10 years. Our objective was to perform visual tasks and explore significant patterns.

We merged data at all time steps and built a *SuperGraph*. Then the community detection and force-directed layout algorithms were performed. Figure 4.15 shows the result. There were 17 communities in total, and they provided a reference for the graph partitioning. At each time step, an extracted community had an equal or smaller size than the

corresponding community in SC. In Figure 4.15, we show the correspondence by labeling communities with indices from 0 to 16. We made a comparison between our SL-based layout and the layout obtained by independent calculations at each timestamp. For the former, identical communities such as *community 1* remain at the same locations during the animation. However, community positions are quite random for the latter. Therefore, our method is better at preserving the mental map.

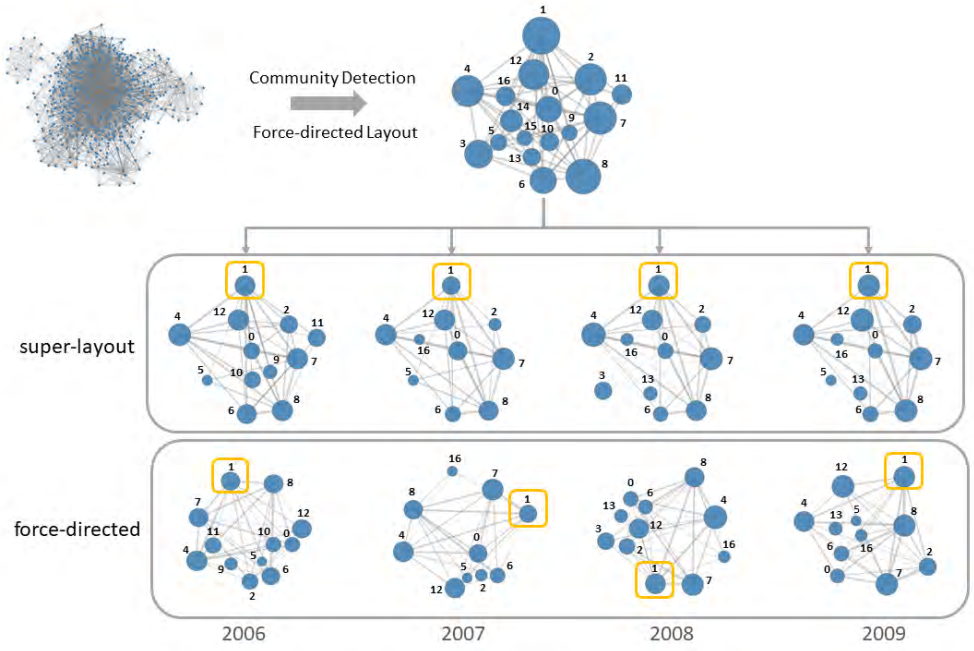


Figure 4.15: Comparison of the layout generated with and without *SL*. The *super-layout* row consists of layouts extractd from *SL*. Layouts in the *force-directed* row are calculated independently.

Figure 4.16 shows the snapshots in four years. The first row consists of the force-directed drawings of the original graph. The second row lists the frames of the animation generated by our method. Due to the stable membership of communities and the static

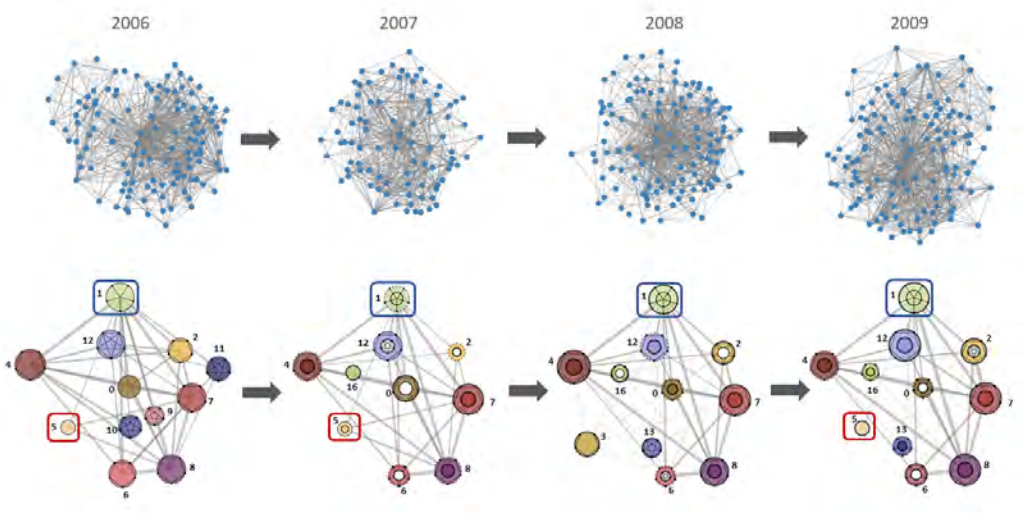


Figure 4.16: Comparison of the original graph drawings (first row) and the simplified drawings with topological pattern glyphs (second row).

positions of identical communities, the animation was quite smooth. Factors that caused visual variations included the connections between communities, the size changes of communities, and the transformation of topological patterns. We assumed that the year 2006 was the beginning point. Hence, all communities had no previous patterns, and the inner circles were empty.

We evaluated the visualization by carrying out the tasks proposed in Section 4.5. By observing the original graphs, we found the year 2007 had the least number of authors. Correspondingly, in the second frame of the animation as shown in Figure 4.16, there are fewer communities and sparser connections than those in other frames (**T1**). We also noticed that the connectivity of communities was more complicated and denser in the year 2006. From year 2006 to 2007, *community 9, 10, 11* disappeared. Though a new *community 16* appeared, most of the communities (i.e., *1, 2, 4, 6, 8*) experienced a size decrease

(**T4**). By measuring the community size, the number of connections to others, and the total weight of connections, *community 1, 4, 7, 8* were believed to be more important (**T2**). They stayed for four years and barely had a contraction. Besides, their topological patterns remained unchanged (**T3, T4**). Conversely, *community 5* had a relatively small size. It made no sense to classify its structure. Hence we did not draw the pattern glyphs and treated *Community 5* as a potential anomaly. After looking into the statistics, we discovered that only one author belonged to this community. We deduced that this author did not have close cooperation with others. By inspecting the community structures and their temporal evolution, we found that *community 2, 6* were relatively dynamic because their patterns changed at every step. Most of the time, their patterns switched between *chain* and *clique*, implying that the collaboration between authors changed dramatically from sparse to dense. In fact, *chain* and *clique* were the two most frequently appeared patterns in this case. It is worth noting that *community 1* was persistent and it kept the *egocentric* pattern (**T3, T4, T5**). So we looked into the inside authors and found that it always contained the *ego* node. Given the characteristic of egocentric networks, it is not difficult to explain the existence of *community 1*.

We obtained the findings mentioned above by observations, and we could confirm them by investigating the statistics of communities. Usually, there were 10 to 12 communities in each year (**T1**). Figure 4.9 (D) shows that, in year 2006, 5 communities had the *loop* pattern and 4 communities had the *clique* pattern. Only one community had the *egocentric* pattern (**T5**). By comparing Figure 4.9 (A) and Figure 4.10, we discovered

that most of the communities in the DBLP dataset were more dynamic than those in the Wikipedia dataset, because the change centrality curves of the former are more fluctuant than those of the latter.

4.7 Discussion

Although our method is effective at visualizing large dynamic graphs and displaying their structural evolution, there is still room for improvement.

To help users identify the community changes, we provide multiple visual encodings that indicate the size variations and the structural transformations. Encodings can be combined to show the concurrency of different types of changes. However, when dozens or even hundreds of communities simultaneously appear on the screen, users will be overwhelmed in their efforts to capture and interpret all these encodings. One potential solution is to control the number of communities, which can be achieved by hierarchical clustering methods. Alternatively, we can use other graph partitioning methods rather than detecting communities based on modularity maximization. For example, METIS [144] divides graphs into a designated number of components while minimizing the edge cuts.

The foundation of our method is the construction of a SuperGraph. We achieve smooth animations and reduce the time cost by conducting calculations at the global level. However, our method cannot guarantee the preservation of the optimal community structures, because we compute communities by extracting subsets from SC rather than by considering the unique connectivity at individual time steps. To obtain better community struc-

tures, we can apply the *Temporal Trade-off* method [88]. But such method produces an incontrollable membership of communities which may decrease the smoothness of visualizations.

We now utilize change centrality and NetSimile to quantify the difference between graph structures. The statistics are displayed along with the simplified visualizations. In this way, users can conduct both visual explorations and numeric investigations.

4.8 Conclusions

We build a visual system for presenting the structural evolution of large time-varying graphs. A simplified form of graph drawing is adopted and a single node now represents a community instead of a vertex. We also improve the layout consistency by conducting calculations on a SuperGraph. Our method makes the visual understanding of complex graph structures feasible after mapping them to the topological patterns. Based on the transformable glyphs, users can easily identify the structural changes from the collapsed node-link diagrams. In addition, visual encodings make it convenient to compare the structures of different communities. For the future work, we plan to apply and compare different metrics to quantify the differences between graph structures. We also want to encode information concerning the classification uncertainty into the visual representations.

CHAPTER 5

VISUALIZING REGION DYNAMICS THROUGH A GEO-SEMANTIC GRAPH BASED METHOD

In this chapter, we extend the foregoing techniques of visualizing dynamic networks to the geographical domain. By using graph structure to model the relationship of data, we can map the problem of finding functional regions to the problem of detecting graph communities. Details about the problem transformation can be found in Section 5.3 and Section 5.4. We also establish a visual system to facilitate region analysis. Visual designs are introduced in Section 5.6.2. In Section 5.7, we evaluate the effectiveness of our method by two real applications and a user study.

5.1 Introduction

Understanding the dynamics of urban activity in large cities and highly developed areas can be a daunting task due to a large number of parameters and the complexity of interactions between structure, accessibility, and the movement of people and vehicles. This complexity increases substantially when additional time constraints are imposed as competing agendas for resource allocation arise and fade over shorter time periods. In order

to reduce this complexity in urban analysis, we propose to segment the larger areas into smaller components and focus on their dominant functional units, one at a time.

There are various segmentation criteria available. For example, based on physical characteristics, the surface of our planet can be largely divided into continents and oceans. As people predominantly shape the large features of our planet, we can think of administrative divisions, transportation, tourism, restricted areas, and so on. For consistency, we use the term *region* to denote non-overlapping sub-areas, and *venue* to indicate the location of designated social activities. The feature that a region carries is called a *function*. Functions reflect the context of human activities.

Region segmentation allows for the overall understanding of the influence or action that a specific function has over a certain geographical area. From this segmentation, urban planners can learn about land uses [145], retailers can decide where to set up advertisements and warehouses [146], and law enforcement officers can rank locations by their security risks [147].

Existing researches on region segmentation rely on various approaches and data sources. Yuan et al. [148] treat the space separated by main roads as regions, and they require extra information like the statistics of POI (Points of Interest), to interpret region functions. However, since road networks update slowly, their segmentation can hardly demonstrate changes of region shapes in real time. Wu et al. build a system called Mobiseg [149] that splits regions by interactively clustering local districts that have similar activity patterns. The patterns are extracted from mobility data, and they vary across time.

Hence, the segmentation can be updated accordingly. But, Mobiseg lacks the exposure of the relationship between regions at different time steps, and does not differentiate between types of regional transformations. This makes it difficult to analyze the evolution of various regions and their functional characteristics over time.

In this chapter, we describe the implementation of a visualization system for presenting and exploring region evolution. The system consists of two collaborating views, the *overall view* and the *comparison view*.

The *overall view* supports the interactions on the entire map. It shows the region centroids at different time steps. From time t to $t + 1$, our matching algorithm aims to find pairs of centroids and their successors that belong to identical regions. Then, the migration trajectories of centroids are displayed by animated sequences. We also demonstrate evolutionary patterns obtained by summarizing all trajectories. From the patterns, we may find that some regions only exist at one time step, which promotes our conjecture that some temporary events reshape these regions but do not have a large impact over the evolution of the regions. Some regions involve only small changes and their centroids shift within a limited area. It is likely that these regions contain stable infrastructures like educational institutes and commercial buildings.

The *comparison view* consists of multiple snapshots from which users can observe region territories. We calculate territories as the minimum geographical area that covers closely located venues where similar human activities occur. Links between snapshots connect matched pairs of region centroids, and they pass through a stack chart that

shows the statistics of the region transformations. We categorize transformations into seven types: *appear*, *disappear*, *expand*, *shrink*, *remain*, *merge*, and *split*, all indicating changes in territory. Topic changes can be identified by comparing keywords.

The *overall view* and the *comparison view* are complementary to each other. The former outperforms the latter in visual scalability. Users can access long-term evolution of regions and roughly judge their stability. The latter integrates multiple visual techniques, and it is more suitable for comparing tasks and inspections of details of an AOI (Area of Interest).

Our main contributions are as follows:

- **Region delimitation**: we put forward a graph-based method to delimit functional regions. Based on geo-textual data, we construct graphs that encode the information about the density of population and the consistency of activities. We take communities detected from the social graph as the reference for calculating region territories.

- **Evolution interpretation**: we propose a matching algorithm to expose region transformations between consecutive time steps. Transformations are categorized into seven types. The evolution of a region can then be interpreted by a series of transformations in chronological order.

- **Evolutionary patterns**: we obtain migration trajectories by connecting region centroids, and abstract them as graphs. By clustering graphs of similar topologies, we reduce the diversity of trajectories and reveal representative patterns of evolution.

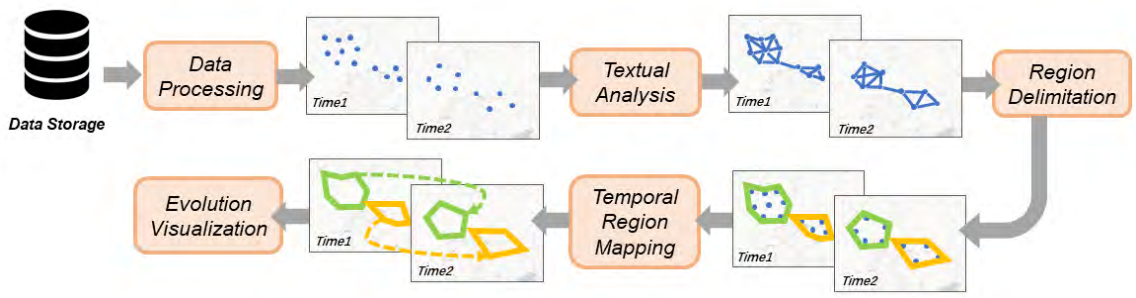


Figure 5.1: The overview of the visual system implementation.

- **Visualization system**: we represent the evolution of a region by using a rich set of visual designs. Animations show coherent transformations, and snapshots depict different region segmentation across time. We evaluate the system by two application scenarios and a study on users' performance of conducting analytic tasks.

5.2 System Overview

As shown in Figure 5.1, we build latent graphs on discrete geographical venues. Close venues are linked. Through textual analysis, the link weight is measured by the similarity score of semantics of end venues. In the procedure of region delimitation, dense venues that also have similar semantics are grouped to form disjoint regions. Applying above steps repeatedly to each snapshot of geo-textual data, we can get independent region distributions. Furthermore, the trend of evolution is exposed by mapping regions at different snapshots. Details can be found in Section 5.3 and Section 5.4.

5.3 Region Delimitation

Our objective is to separate regions that reflect the natural gathering of people and distinguish from each other by the theme of inside activities. Density-based algorithms like DBSCAN can be used to cluster venues by their spatial closeness, but the corresponding activities in one cluster do not necessarily have similar semantics. Many previous work employed mobility data such as taxi routes and commuting flows [150, 151, 149]. However, extra information like POI statistics were needed for region interpretation. We propose a graph-based method. It encodes the spatial and semantic proximity of venues simultaneously. With geo-textual data, we can directly learn topics to characterise regions.

5.3.1 Data Description

We use a four-tuple $< time, venue, text, meta\_data >$ to depict a geo-textual item, where $time$ and $venue$ denote when and where an activity happens and $text$ describes the context of the activity. A $venue$ is identified by a coordinate of longitude and latitude, and $text$ consists of a list of words. $meta\_data$ include supplementary information, such as a photo taken at the venue. During the pre-processing stage, we aggregate data items by uniform time intervals, e. g., day, month or year, for the ease of analysis.

5.3.2 Semantics Extraction

The words that people use to describe activities are highly diverse and are likely to be meaningless, such as stop words. We reduce the difficulty of analysis by extracting in-

sightful semantics and then calculating their similarities. Specifically, the LDA (Latent Dirichlet Allocation) model is adopted, because it returns interpretable topics distributed at all venues.

LDA represents a document as a probabilistic mixture of topics learnt from the corpus. A topic corresponds to a cluster of words. At one time step, the *corpus* is composed of words collected from the text component of all geo-textual items, excluding meaningless words. The text component of each data item is regarded as a single *document*. Assuming that there are n topics $\{t_1, t_2, \ldots, t_n\}$ in the corpus, we compute:

- ***Venue Topics***: the topic distribution at a venue v, denoted by $T_v = [p_{t_1}, p_{t_2}, \ldots, p_{t_n}]$. $\sum_{i=1}^{n} p_{t_i} = 1$, where p_{t_i} is the probability of topic t_i.

- ***Region Topics***: the average topic distribution of all venues within a region r, represented by $T_r = \frac{1}{|V_r|} \sum_{i=1}^{|V_r|} T_i$, and $|V_r|$ is the number of venues inside r.

For venues that associate with multiple items, T_v is an averaged topic distribution of these items. To update topics, we conduct LDA repeatedly for each time step.

We can measure the extent of consistency of activities in r by the information entropy of T_r, which is defined as:

$$E_r = -\sum_{i=1}^{n} \overline{p_i} \log \overline{p_i}, \tag{5.1}$$

where $\overline{p_i}$ is the averaged probability of topic t_i of all $|V_r|$ venues. A low entropy value implies that the region tends to have an explicit function that can be interpreted by a

dominate topic in T_r. Otherwise, we can find various types of activities involved in the region.

5.3.3 Geo-Semantic Graph

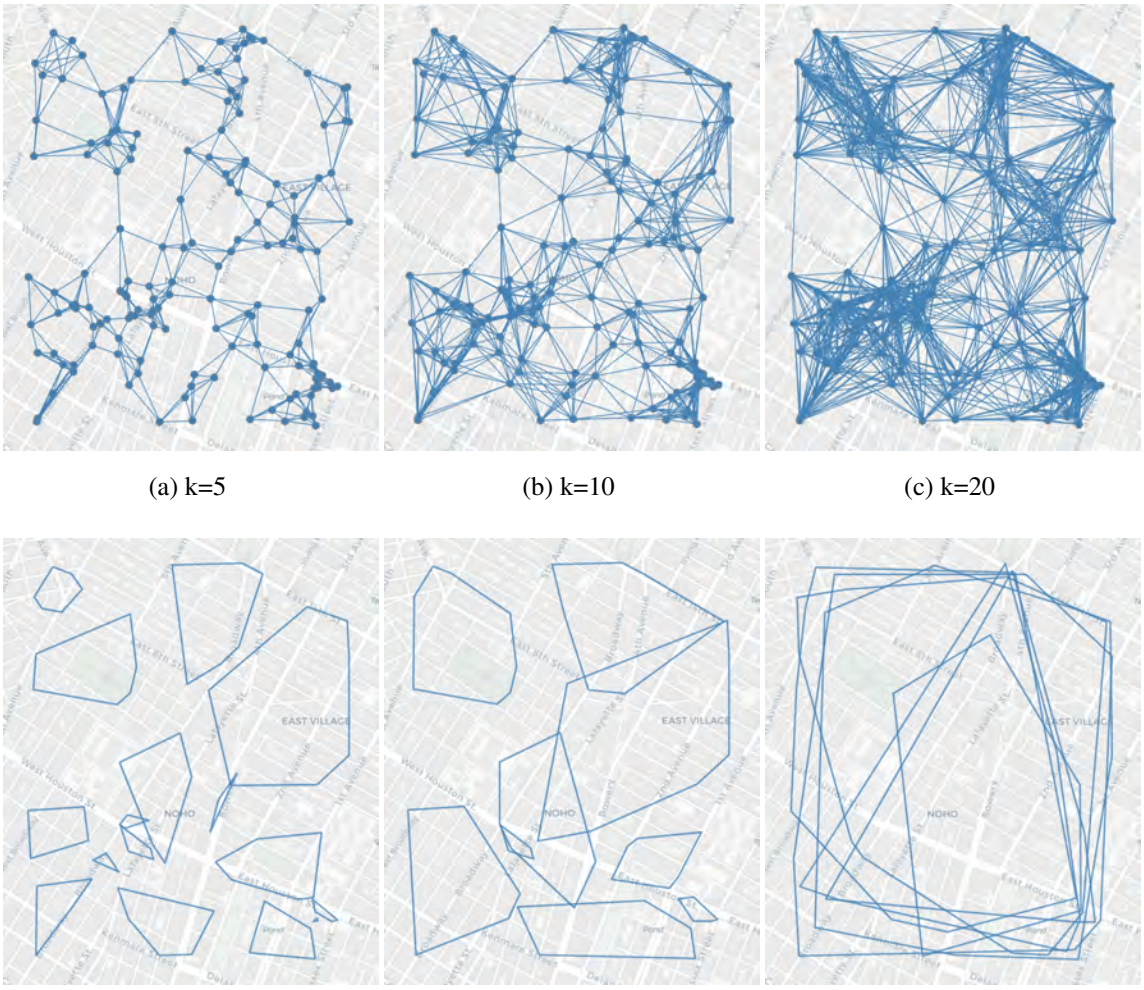
We abstract venues as graph vertexes and each venue only connects with its k nearest neighbors. The weight of connections is equal to the similarity score between topic distributions of two end venues v_i and v_j, which is defined by the reciprocal of symmetric Kullback-Leibler divergence:

$$KL(T_{v_i}, T_{v_j}) = \sum_{t=1}^{n} T_{v_i,t} \cdot \frac{T_{v_i,t}}{T_{v_j,t}}, \tag{5.2}$$

$$KL_{sym}(T_{v_i}, T_{v_j}) = \frac{1}{2} \cdot (KL(T_{v_i}, T_{v_j}) + KL(T_{v_j}, T_{v_i})). \tag{5.3}$$

Then, we cluster venues by applying the Louvain method [67] which aims at maximizing the graph modularity. Modularity measures the ratio of connections between venues in same clusters to connections between venues in different clusters. Initially, to represent region territories, we encompass venues belonging to same clusters with convex hulls. Later in Section 5.6.2, we refine visual aesthetics by tiling convex polygons with hexagonal grids.

An example is shown in Figure 5.2. Given the same set of venues, we can generate graphs with different k values. Normally, a lower value results in a sparser graph and more regions. Besides, regions are relatively smaller and less overlapped. With the increase of



(a) k=5 (b) k=10 (c) k=20

(d) k=5 (e) k=10 (f) k=20

Figure 5.2: Given an identical set of venues, the latent geo-semantic graphs (first row) and the obtained regions (second row).

k, each venue gets connected to more neighbors that are further away. An extreme case is that the graph is fully connected, and we get only one region covering all venues.

Based on the way of graph construction, closely located venues are more densely connected, as we can see in the first row of Figure 5.2. Hence they are more likely to be in the same cluster. Besides, if a link has a heavier weight, then its incident vertexes are also

more likely to be assigned to the same cluster. In summary, we consider both the density of population and the consistency of activities to separate regions.

The graph is latent because it is invisible to users. We repeat the graph construction for individual time steps to update the region segmentation. Venues are organized into a k-d tree to speed up the search for nearest neighbors. Due to the sparsity of data at specific areas, a venue might be far away from its neighbors. In case of getting large regions with few venues inside, we can remove those connections whose end-venues have a geographical distance that exceeds the limitation.

Besides the Louvain method for clustering graph vertexes, a lot of other approaches are also available [117], such as the traditional partitioning methods [152, 153], spectral clustering methods [154], etc. The Girvan and Newman algorithm is the first modern technique of community detection. It is implemented by iteratively removing links based on their betweenness. However, it has a high computational complexity ($O(n^3)$ on a sparse graph, and n is the number of nodes). We chose the Louvain algorithm because of its high efficiency in processing large graphs. It runs in time $O(n \log n)$. Also, it can find communities that contribute to a high modularity without knowing their count in advance. According to the comparison made by Lancichinetti and Fortunato [66], the Infomap [69] and the multi-resolutional method [155] can be alternatives that have a similar performance.

5.4 Region Matching

To understand how a region evolves from time t to $t+1$, we need to find its target regions after transformation, and then decide the type of transformation. Hence, we propose a matching algorithm as shown in Algorithm 1. The output are two sets of matching results S and S' for regions at t and $t+1$, respectively. M and M' denote the set of matched regions. By applying matching operations to a sequence of snapshots, we can track the temporal changes of regions.

For two regions to be matched, they need to fulfill the following conditions: first, their areas overlap; second, they should have a certain proportion of designated venues inside the overlapping area. The second condition refines the matching result by inspecting the importance of the overlapping area. The importance depends on the density of venues rather than the area size, because a large area containing only a few venues is perceived to be less important.

In *line 2*, we firstly build a k-d tree in $O(n \log n)$ time to organise centroids of all regions in R_t and R_{t+1}, and n is the total number of regions. Then, searching the nearest region is an $O(\log n)$ operation on average. In *line 4-5*, we calculate o by adopting the Weiler-Atherton clipping algorithm [156]. In *line 6-7*, we use the ray casting method [157] to determine if venues are inside o. Increasing the threshold in *line 8* will decrease the number of matched regions.

To accelerate the matching procedure, we reduce the candidate targets twice. First, in

line 2, r_i is only compared to its k nearest rather than all regions in R_{t+1}. Because the probability that two regions are overlapped decreases when their distance increases. In our tests we find that a region rarely overlaps with more than 6 other regions, i.e., $k = 6$.

Algorithm 1 Matching Regions

Input:

R_t, R_{t+1}: the set of regions at time t and $t+1$, respectively; $R_t = \{r_1, r_2, \cdots, r_m\}$, $R_{t+1} = \{r'_1, r'_2, \cdots, r'_n\}$.

Output:

$S = \{(r, M_r) | \forall r \in R_t, M_r \subseteq R_{t+1}\}$;

$S' = \{(r', M'_{r'}) | \forall r' \in R_{t+1}, M'_{r'} \subseteq R_t\}$

1: **for** $r_i \in R_t$ **do**
2: $NR_i \Leftarrow k$ nearest regions to r_i in R_{t+1};
3: **for** $nr_i \in NR_i$ **do**
4: **if** $Overlapped(r_i, nr_i) =$ ***True*** **then**
5: $o \Leftarrow$ overlapping area of r_i and nr_i
6: $v_i \Leftarrow$ venues within r_i and also locate in o;
7: $nv_i \Leftarrow$ venues within nr_i and also locate in o;
8: **if** ($\frac{|v_i|}{|r_i|} > threshold$) and $\frac{|nv_i|}{|nr_i|} > threshold$) **then**
9: add nr_i to M_{r_i};
10: add r_i to M'_{nr_i};
11: **end if**
12: **else**
13: break;
14: **end if**
15: **end for**
16: **end for**

The distance between regions is measured by the distance between their centroids that are obtained by averaging all venue coordinates. Second, we sort regions in NR_i in ascending order based on their distances to r_i. In *line 12*, the matching process stops once a candidate does not overlap r_i. It is noteworthy that acceleration might compromise the matching accuracy in that certain regions probably overlap with more than k neighbors. Also, we should have used the single-linkage criteria [158] to measure the distance between region borders, but it takes too many calculations.

The algorithm is applicable to any region sets, no matter they are adjacent in time or not. Matching results can be categorized into five different types, and their relationship to region transformations are as follows. For the ease of explanation, we use s_1 and s_2 to denote the former and latter set of regions, respectively. Region r_1 belongs to s_1 and r_2 is in s_2.

- ***One-to-Zero***: no regions in s_2 match with r_1. r_1 *disappears*;
- ***Zero-to-One***: no regions in s_1 match with r_2. r_2 *appears*;
- ***One-to-One***: only r_1 matches r_2 in s_1 and vice versa. The region territory may *expand*, *shrink*, or *remain*;
- ***One-to-Multi***: r_1 matches with more than one region in s_2. r_1 is *split* into multiple regions;
- ***Multi-to-One***: r_2 matches with more than one region in s_1. Multiple regions *merge* into r_2.

5.5 Evolution Patterns

We create migration trajectories based on the temporal matching results which can be obtained by repeatedly applying matching operations to regions at every two consecutive time steps. A trajectory represents the connections between centroids of a region, but embodies no information about the differences in size and function.

Many regions often undergo similar transitions of evolution. This can be exposed via representative evolution patterns by abstracting trajectories as graphs. Region centroids are then mapped to graph vertexes which are stamped by their time steps. Graph edges indicate the original connections between centroids. Figure 5.3 shows an example. R_1 and R_2 have only *one-to-one* transformations, hence their corresponding graphs have the same topology. Two transformations, a *split* and a *merge* transformation, are applied to R_3, so there is a cycle in the graph.

Graph abstraction simplifies trajectories by ignoring details such as the spatial location of centroids and the distance between them. Then, we can easily find regions that undergo similar evolution procedures by clustering abstract graphs based on topology. Netsimile [159] is an approach for quantifying the similarity between size-independent graphs. It firstly converts graphs into signature vectors by aggregating topological attributes of vertexes, including degree, clustering coefficient, etc. Then, we can apply a standard clustering method (e.g., k-means) on these vectors with the cosine distance. Evolution patterns are revealed by randomly choosing a representative graph from each cluster.

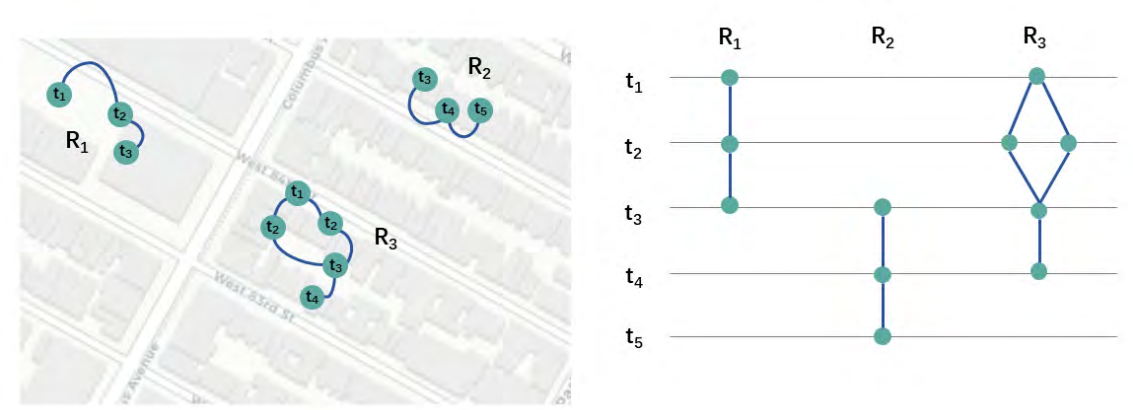


Figure 5.3: Conversion from migration trajectories to graphs. Nodes represent region centroids and they are stamped with time steps.

5.6 Visualization

We build a visual system to facilitate the exploration of region evolution. The system consists of an *overall view* where users can access details of venues, like their descriptions and distributions, and a *comparison view* that displays results of dynamic region segmentation.

5.6.1 Requirement Analysis

Our region delimitation method is based on discrete venues, along with their contextual information. Therefore, we focus on representing both low-level (i.e., venues) and high-level (i.e., regions) geographical elements. Their visualization requirements are as follows:

- ***Venues***: mark their locations on the map, and allow users to interactively select

venues and investigate the meta data, such as time, photos, comments, etc.

- ***Regions***: improve aesthetics of the representations, and make regions visually comparable.

Based on the visual task model proposed by Andrienko et al. [160], our system provides *static* and *dynamic* information. At a specific time step, static information includes:

- ***Venue distribution*** (**S1**): Which part has a higher density of venues within the focal area? Where is the centroid of venues?
- ***Region distribution*** (**S2**): How many regions does the focal area possess? How large is the territory of a region? What is the relationship between the region segmentation and the venue distribution?
- ***Region semantics*** (**S3**): What is the entropy level of region topics? Does a region have an explicit function? What are the regions that have similar functions?

During a period, dynamic information includes:

- ***Territory transformation*** (**S4**): How does a region territory change over time? Does it remain stable, become larger/smaller, or even split/merge into other regions?
- ***Semantic transition*** (**S5**): What is the difference of topic distribution between consecutive time steps, regarding individual regions?
- ***Area dynamics*** (**S6**): How many regions contribute to each type of transformations, within an area of interest?

5.6.2 Visual Design

Figure 5.4 shows the interface of the visual system. We have two collaborating views, each characterised by animations and small-multiple representations. The *overall view* provides both static and dynamic information at a global level. It plays animations of the movement of region centroids. Animations are effective in solving the scalability issues of visualizing temporal data. However, users may suffer from visual clutter due to trajectory crossings [161]. Also, it is difficult to show region territories at different time steps on the same map because of visual overlaps. Therefore, we allow users to select an AOI and perform further explorations in the *comparison view*. A sequence of snapshots are juxtaposed along the timeline, each exposing region details at a specific time step. Adjacent snapshots are connected by links that demonstrate the matching relationship between regions.

Overall View

In this view, we provide utilities for users to access low-level static information. A scatter plot is used to display the geographical distribution of discrete venues and a heatmap shows the density distribution. Our concerns of visualizing high-level information are: first, what is the aesthetic way to illustrate regions that are derived by applying the delimitation method introduced in Section 5.3; second, how to make region evolution straightforward to users.

Using convex hulls to represent regions is not a good option, because their irregular

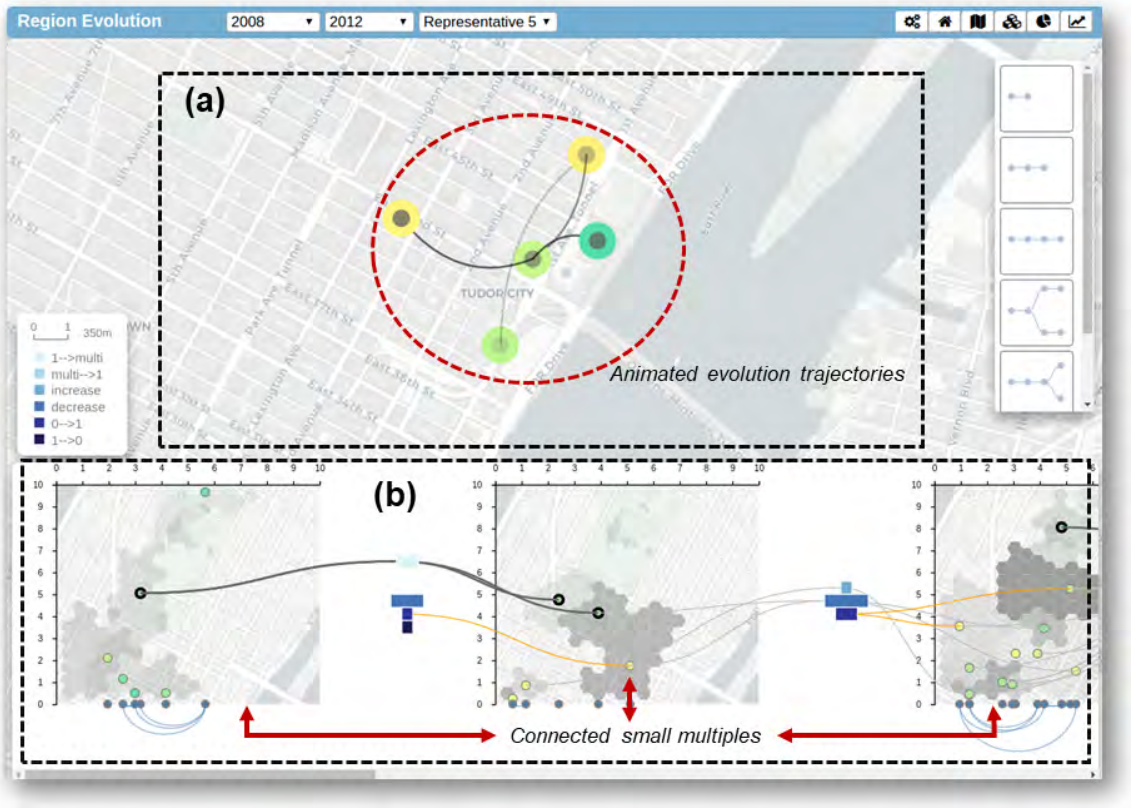
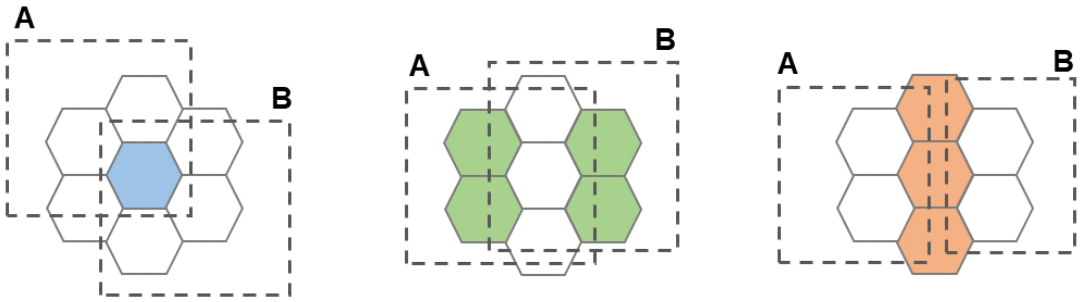


Figure 5.4: Interface of the visual system. (a) Overall view: plays the animation of region evolution; (b) Comparison view: contains snapshots of region distributions in an AOI.

shapes are hard to compare and they might overlap slightly at borders. The overlapping is caused by peripheral venues, and we later measure their importance to decide if they should be treated as outliers. In this work, we adopt the tiling techniques [162, 163] to convert convex polygons to hexagonal grids. Besides, we achieve non-overlapping representations by exclusively assigning grid cells to a region.

Region tiling facilitates visual comparisons by making the territory size countable, but it is difficult to preserve the boundary of convex polygons and minimize the time cost of



(a) (b) (c)

Figure 5.5: Problematic situations of assigning cells at the borders of regions. (a) The hexagon (blue) is inside both region A and B; (b) Hexagons (green) are inside one region and intersect with another; (c) Hexagons (orange) intersect with both region A and B.

tiling. In general, smaller hexagons can be aggregated into arbitrary polygon shapes while maintaining uniform convexity internally and the topic entropy in one cell is lower. But the time cost may increase as more hexagons need to be calculated for the same region.

The primary operation of tiling is using the ScanLine method [164] to determine if a cell completely resides in a region polygon. The procedure can be accelerated by indexing grid cells with an r-tree. Meanwhile, we have to deal with some ambiguous situations at boundaries. As shown in Figure 5.5, cells might intersect with or belong to multiple polygons simultaneously. Let $\mathcal{R}$ be the set of regions involved with a cell c. For $r \in \mathcal{R}$, v_{rc} is the set of venues inside the overlapping area of r and c. $|r|$ is the total number of venues in r, and c_r is the centroid of r; $dist(\cdot)$ returns the distance between two venues.

The contentious cell c will be allocated to the region which satisfies:

$$r = \arg\max_{r \in \mathcal{R}} \frac{1}{|r|} \sum_{v \in v_{rc}} \frac{1}{1 + dist(v, c_r)}. \tag{5.4}$$

Equation 5.4 implies the impact that venues inside the overlapping area have on the whole region. If the impact is negligible, we skip c as a part of the tiling. Also, for reducing holes in the tiling, we prefer to assign c to regions that totally cover rather then intersect with c.

Figure 5.6(a) shows an example of the original polygonal representations of regions, and in Figure 5.6(b), we can see the tiling result based on the strategy defined in Equation 5.4. It is worthy noting that, in Figure 5.6(b), cell **A** is inside r_6 and intersects with r_8, and it is finally allocated to r_6. Cell **B** intersects with both r_5 and r_6, and it finally goes to r_5. Cell **C** intersects with r_4, but its inner venues have weak impact on r_4, hence it is discarded.

Alternative geometries include squares and triangles. In Figure 5.7, we compare the three types of tiling given an identical region. Single cells of each grid have the same size, and the ratio of the side length of cells is approximately 24:15:10. According to Chen et al [165], the triangle grid introduces visual ambiguities, because it consists of triangles oriented in two directions. The square grid brings illusions of stretching along the vertical and horizontal levels. Hence, we choose the hexagonal grid for obtaining better aesthetics and visual compactness.

An animation that shows the movement of region centroids helps users to understand

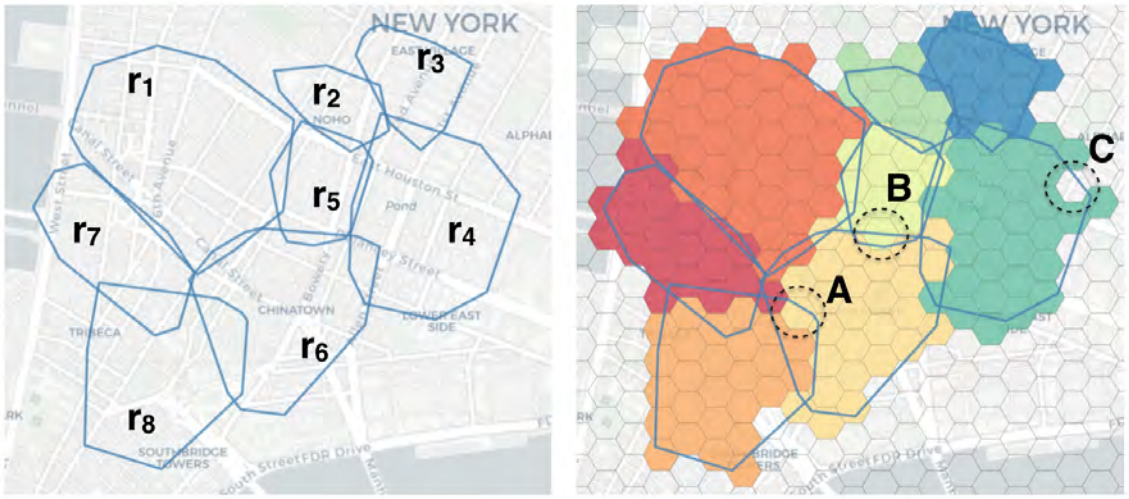


Figure 5.6: Region tiling with hexagonal cells. (a) Polygonal region representations; (b) Hexagonal tiling. Different regions are distinguished by using a ColorBrewer platte [132].

Figure 5.7: Tiling a region by (a) triangles, (b) squares and (c) hexagons.

the region evolution. We take centroids as the delegates of regions, because directly drawing region territories causes visual overlaps on a single map, like the example in Figure 5.8. Users are still allowed to inspect the underlying territory by clicking a centroid.

We map the chronological order to the opacity of inner circles of centroids and the curved links. The higher the opacity, the closer the timestamp of a centroid to the current time. Alternatively, arrows can represent the order. But when centroids are close to each other, the links are short and inserting arrows to such kind of links increases the visual clutter. The default color of inner circles and curved links is *black*. Outer circles of centroids are filled according to the linear gradient of color shown in the top-left corner of Figure 5.8. The warmer the color, the higher the topic entropy of the region in all regions.

Comparison View

In the *overall view*, users can select a rectangular AOI by dragging. Then, they can access and compare the details of multiple snapshots simultaneously in the *comparison view*, without being interrupted by the visual clutter caused by territory overlaps. Figure 5.9 shows an example of the *comparison view* that contains two snapshots, each representing the region delimitation result at a specific time.

We use various visual designs to encode the statistics of venues and regions. Thick lines highlight the boundaries of regions. The opacity of region fillings indicates the number of inner venues, and the default color is *black*. The darker the filling, the more venues the region contains. Heatmaps located at the horizontal and vertical axes reflect the distribution of venues along the longitude (*green*) and latitude (*magenta*) directions, respectively. Besides, the graduated axes are helpful in calculating the migration distances. Users can switch between different regions by clicking their centroids which are drawn as

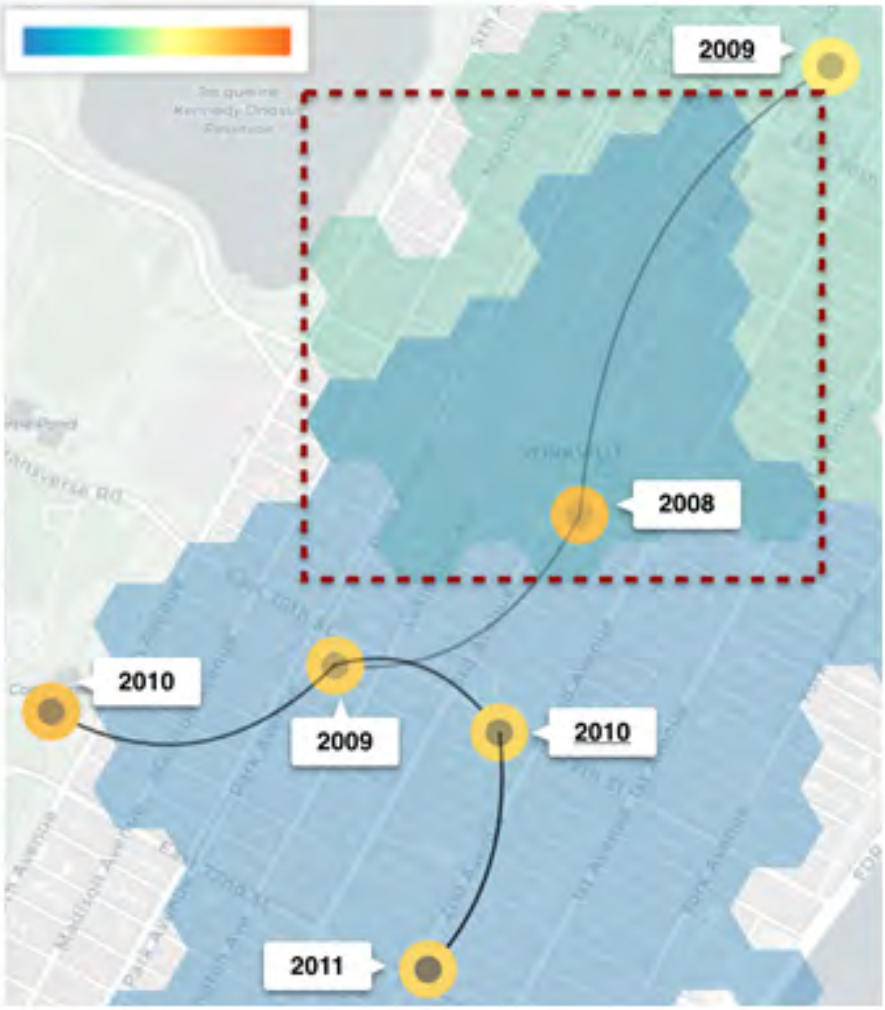


Figure 5.8: The migration trajectory of a region. Each centroid is denoted by two concentric circles. The opacity of inner circles indicate the chronological order and the color of outer circles reflect the entropy level of region topics. Territories in the year 2009 and 2010 are represented by blue and green hexagonal grids, respectively, and their overlapping area is marked by a dashed rectangle.

small circles. The circle fillings reveal the entropy level of region topics and they follow the same color scheme adopted in Figure 5.8. An arc diagram at the bottom of each snap-

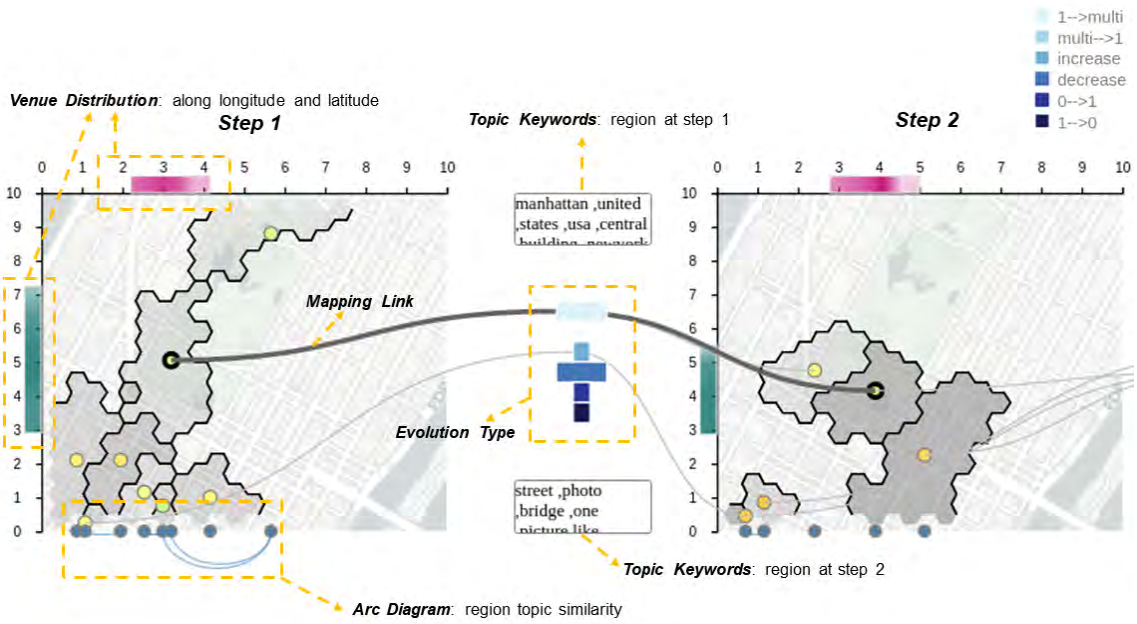


Figure 5.9: Two snapshots from the comparison view, each displaying the distribution of regions at a specific timestamp. Snapshots are connected by links showing the matching results of regions.

shot depicts the similarity of topic distributions between each pair of regions. Nodes in the diagram are projections of centroids to the bottom axis. The range of similarity values is $[0, 1]$, and users can decide the number of visible arcs by controlling the threshold.

Adjacent snapshots are connected by links whose end points are centroids of matched regions. Then, it becomes convenient to track the temporal transformation of regions. Links are categorized according to different types of matching results defined in Section 5.4. Among them, the *One-to-One* type is divided into *increase* and *decrease* categories, indicating the size change of regions. The six categories are denoted by stacked rectangles whose width is proportional to the number of regions belonging to that cate-

gory. We note that a link is invisible if any of its end centroids goes out of the AOI. When clicking on a link, it is highlighted with a thicker stroke. Meanwhile, the keywords of topics of two end regions are listed separately in the boxes that are located at the top and bottom space between snapshots.

5.7 Evaluation

We demonstrate the effectiveness of our methods by presenting two application cases and a user study. All experiments were conducted on a desktop machine with 2.20GHz, Intel Core i5 CPU, 16.0 GB RAM and 1680×1050 pixel resolution. The visual system was built by using JavaScript and D3, and interactions on the map were supported by Leaflet.

5.7.1 Case 1: Flickr

YFCC100M dataset [166] contains records of photos or videos uploaded by Flickr users during the year 2004 to 2014. In this case, we focused on the period from the year 2008 to 2012 and filtered out the records whose textual descriptions and geo-locations were both available. The focal area was bounded to New York City.

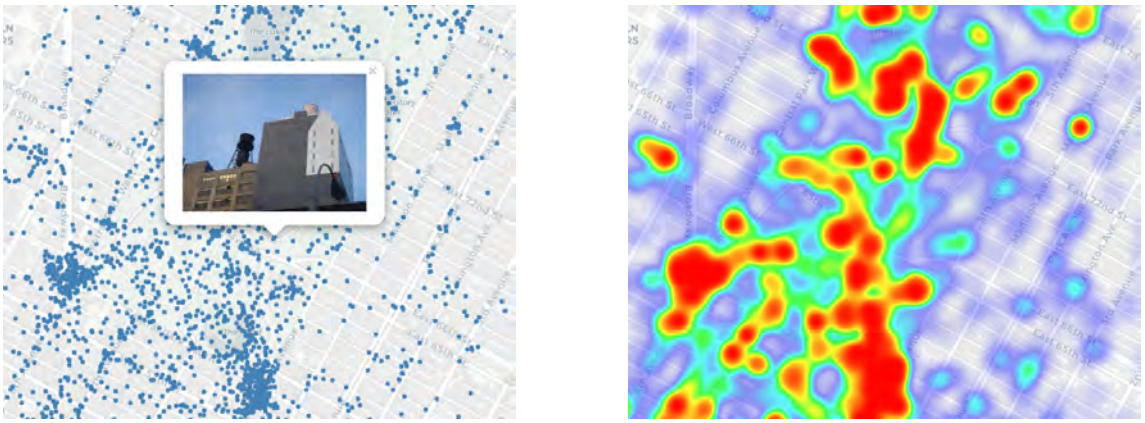
Data items were manipulated to fit the format that we defined in Section 5.3.1. We then aggregated items based on the time interval of 1 year. By doing experiments with different k values, we decided to connect each venue to its 20 nearest neighbors. Consequently, region overlaps were minimized. Statistic samples of the venues and regions can be found in Table 5.1.

	2008	2009	2010	2011
$\|V\|$	26 784	30 771	35 459	43 321
k=5	85 061/384	97 517/465	112 520/502	137 007/586
k=10	166 696/168	190 491/187	220 187/214	268 422/228
k=20	329 901/99	377 000/112	435 585/124	531 500/137

Table 5.1: For the Flickr dataset, given the number of venues ($|V|$) for each year from the year 2008 to 2011, we show the ratio of the number of edges of the latent graph to the number of regions ($|E|/|R|$) for each of the three values of k=5, 10, and 20.

We allowed users to view the information of individual venues. Figure 5.10(a) is a scatter plot of venues. When clicking on a venue, a popup window displays the corresponding media object. Figure 5.10(b) shows a heatmap of the venue distribution. The warmer the color, the denser the venues. We found that most of the blocks in New York City were crowded with Flickr users, especially in midtown and downtown (**S1**).

One significant discovery was that our region delimitation procedure successfully located landmarks in New York City (**S3**). The LDA model extracted 12 topics at each time step. With the help of topic keywords listed in Table 5.2, we identified regions covering landmarks, such as Times Square and The Empire State Building. The result is shown in Figure 5.11. These regions were detected mostly because people tended to add the name of landmarks in textual descriptions. Most of the corresponding evolution trajectories only involved with *One-to-One* transformations. However, the trajectory of the Central Park region experienced a few times of split. An explanation could be that, within the vast area, people were more likely to move around and gather at different locations. Based



(a) (b)

Figure 5.10: Within an area of NYC in the year 2008, (a) a scatter plot in which blue circles denote venues, and (b) the corresponding heatmap that reflects the popularity of different places.

on the topic keywords, we also inferred that the assembly locations could be the baseball fields. We then confirmed the inference by *Google Maps* API.

We set the threshold in Algorithm 1 to 0.6 to get a balanced distribution of different types of transformations. Some regions remained stable across time. Their evolution trajectories did not embody *split* or *merge* transformations. Besides, region centroids shifted within a limited area (**S4**), and the entropy of the topic distributions remained at a low level. We conclude that such regions provide constant utilities and services, like the Museum in Figure 5.11. Conversely, regions without an explicit function were potentially more dynamic and their evolution trajectories contained more types of transformations. By clustering the trajectories, we revealed the evolution patterns and the representatives were shown in Figure 5.12. Patterns 1, 2 and 3 vary in duration. In fact, many regions appeared at only one timestamp, and the trajectory graph contained only a single node.

Topic Index	Keywords	Landmark
t_{10}(41%)	brooklyn, times, bridge, east, art, street, square, manhattan	Times Square
t_4(23%)	st, college, years, day, natural, fair, benoit, amnh, history, american, parade, museum	American Museum of Natural History
t_1(52%)	central, building, park, baseball, stadium, manhattan, city, states, yankee	Central Park
t_{12}(44%)	street, manhattan, state, building, center, avenue, empire, governors, nuev, apple, big	The Empire State Building

Table 5.2: Keywords of the dominating topics in four landmark regions. The first column lists the index and the probability of topics.

Figure 5.12 also displays the spatial distribution of regions that match the patterns 3, 4 and 5 (**S2**). In this case, a majority of regions contributed to pattern 3. Regions that were assigned pattern 4 were more likely to be detected around transportation hubs. A minority of regions went through a combination of transformations, and their evolution trajectories did not match with any of the five patterns, such as the Central Park region shown in Figure 5.11.

We selected an AOI with diagonal corners at [-74.0012, 40.7822] and [-73.9599, 40.7562]. The radius of hexagonal cells was set to 100 meters, and the unit of longitude and latitude axes was set to 350 meters. The *comparison view* shown in Figure 5.13

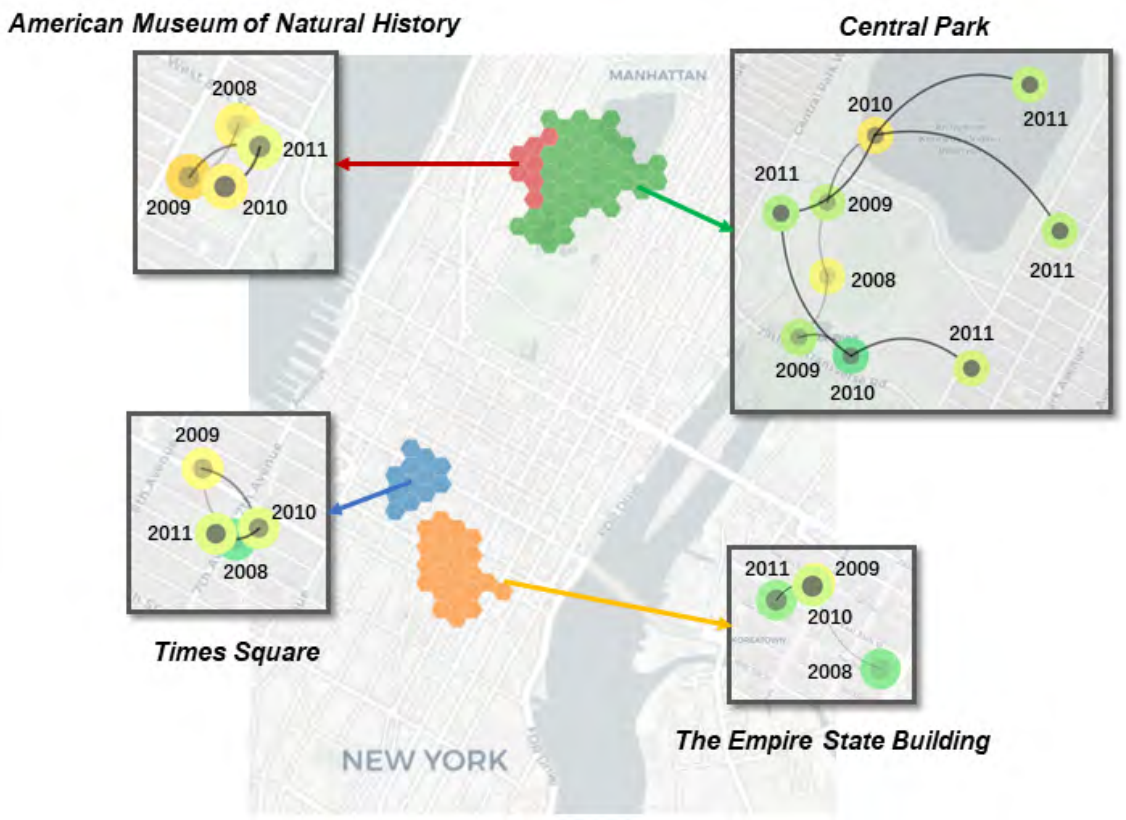


Figure 5.11: Four landmark regions in the year 2008. Their evolution trajectories are shown in the enlarged rectangles.

contains four snapshots from the year 2008 to 2012. Between the first two snapshots, thick links represent a *split* transformation from one region to two regions. The stacked rectangles in the middle space indicate that no regions involve with *merge* and *increase* transformations (**S6**). In fact, the *merge* transformation is missing during the whole period. The *orange* link suggests that a region appears, and it is invisible by default because there is no matched centroid in the first snapshot. Users can inspect the hidden links by clicking on rectangles. Among the last three snapshots, thick links demonstrate the

Figure 5.12: Representative patterns of region evolution. Three map views show the spatial distribution of regions that contribute to pattern 3, 4 and 5, respectively.

evolution of regions in Central Park. From the opacity of fillings, we can see that the popularity of regions does not vary dramatically. The *split* transformation happens twice. One descendant region is beyond the range of AOI in the fourth snapshot. In the last snapshot, the two descendant regions have similar topics and possess a communal segment of boundary. The reason of their separation is probably because region centroids are far from each other, and few venues locate at the periphery. We can confirm the inference of venue distributions by observing heatmaps along the graduated axes (**S1**).

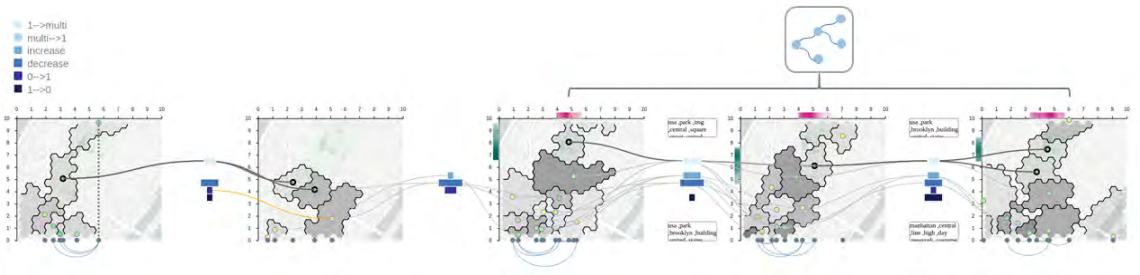


Figure 5.13: Five snapshots of an AOI in NYC. The pattern graph depicts the evolution of regions in Central Park.

5.7.2 Case 2: Yelp

We collected the reviews of businesses in Las Vegas from *Yelp Dataset Challenge* [167]. Data items went through a pre-processing stage. The time interval was one month. Table 5.3 lists statistic samples of the dataset.

There are two notable distinctions between the Yelp and Flickr datasets. First, the temporal distribution of venues in the Yelp dataset is more stable. Because venues in the Flickr dataset are arbitrary places where people may take photos on a whim. However, in this case, business venues are fixed. The difference between venues at different timestamps mainly reflects the establishment of new businesses or the termination of old businesses. Second, the Yelp dataset has less topic varieties as the comments are all about the businesses. Therefore, regions detected in this case are supposed to have stable territories and consistent topic themes over time.

We tested different k values when constructing latent graphs and the final value was

	Jan.	Feb.	Mar.	Apr.
$\|V\|$	6802	6792	7249	7199
k=5	21 709/156	21 581/148	23 211/162	22 948/155
k=15	63 230/57	63 283/57	67 541/54	66 999/57
k=25 $(dist_{nbr} < \infty)$	105 062/38	104 928/41	112 044/40	111 416/40
k=25 $(dist_{nbr} < 1.5km)$	100 949/57	100 962/55	107 728/56	107 368/53

Table 5.3: The number of venues ($|V|$) in the Yelp dataset. When k=5, 15, 25, the ratio of the number of edges of latent graphs to the number of regions ($|E|/|R|$). $dist_{nbr}$ denotes the distance between connected venues.

25. Besides, we compared the region delimitation results when the distance between connected venues was or was not constrained. The heatmap in Figure 5.15 shows the uneven distribution of businesses. We can see that businesses are only crowded in two areas (red color). If there was no distance limitation, venues in sparse areas would find neighbors in far places. Then, we would obtain large regions in low-density areas, such as R_1 and R_2 in Figure 5.14(a). When we set the threshold to 1500 meters, the number of edges in latent graphs decreased as shown in the last row of Table 5.3. In addition, we eliminated the impact of isolated venues that were marked by rectangles in Figure 5.14(b). Tiny regions were ignored. We also found that the distance limitation did not drastically affect the region delimitation in high-density areas marked by ellipses in Figure 5.14.

We compared the region evolution of two AOIs. One was the high-density area *B* in

(a) (b)

Figure 5.14: Convex polygons (red) represent regions that are detected when the distance between connected venues ($dist_{nbr}$) is: (a) not limited; (b) shorter than 1500m.

Figure 5.15. It located at the center of Las Vegas. The other one was the low-density area *A*. *A* and *B* had the same size. Their snapshots at four timestamps are displayed. From the opacity of region fillings, we see that most of the regions in *B* encompass more venues than regions in *A*, even though they have smaller territories. Besides, there are more regions in *B* and the stacked rectangles indicate that these regions involve with higher dynamics that are caused by various types of transformations. However, regions in *A* are relatively stable except the size changes.

A majority of regions in Las Vegas had the same evolution pattern as the representative shown in Figure 5.15. It reflected the stable distribution of venues and regions. However, outlier patterns such as the one in Figure 5.15, were found especially at high-density areas like *B*. Frequent *split* and *merge* transformations occurred to the corresponding regions. Venue descriptions had similar semantics because they were about the characteristics of

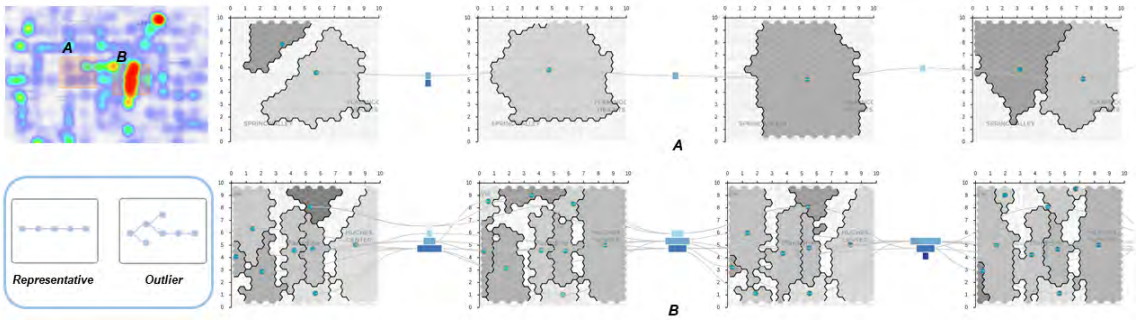


Figure 5.15: *A* and *B* are AOIs of the same size. Venues are sparse in *A* and dense in *B*. Snapshots of *A* and *B* from January to April are displayed in the up and bottom rows, respectively.

businesses, such as the quality of service. Hence, edges of individual latent graphs had similar weights. Region delimitation depended more on the closeness of venues than the proximity of semantics (**S2**). The boxplot in Figure 5.16 represents the entropy of region topics. We extracted 6 topics at each time step. When comparing region topics in the Yelp and Flickr cases, the former have lower entropy levels and smaller variation ranges of entropy values. It implies that Yelp regions have more explicit and consistent topics, while Flickr regions possess multiple topics and none of them plays a dominant role.

Table 5.4 lists the time cost of the main procedures in our method. Areas considered for the Flickr and Yelp cases are NYC and Las Vegas, respectively. The graph construction procedure includes calculations of k nearest neighbors and semantic similarities between venues. The second column shows the running time of our delimitation algorithm. The most time-consuming part of tiling is generating a hexagonal grid for the whole city area, and the tool we use is Turf.

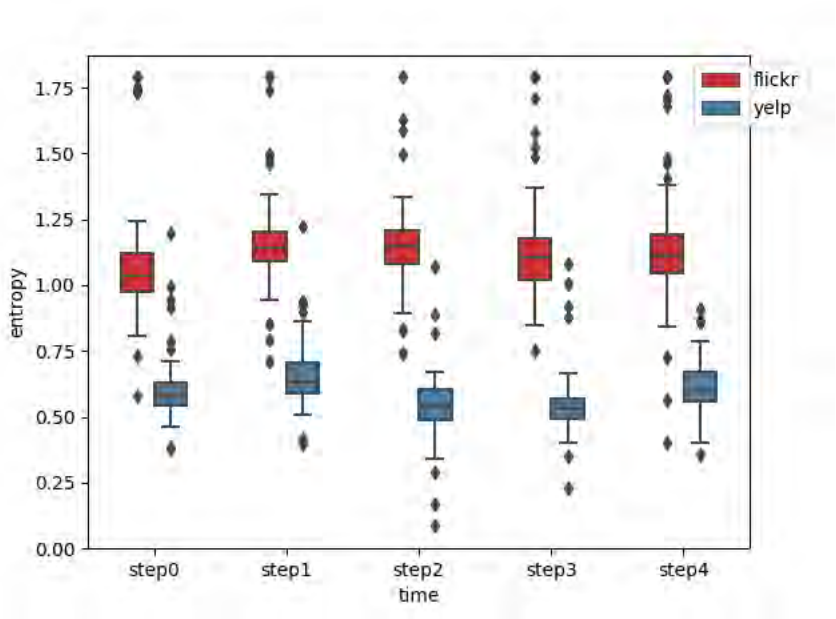


Figure 5.16: Statistics of topic entropy among all regions. Yelp regions have lower-level entropy, implying that they embody more explicit topics.

	Graph Construction	Region Delimitation	Matching	Tiling
Yelp	0.03	0.44	1.01	0.21
Flickr	0.21	2.37	2.49	0.83

Table 5.4: Among all time steps, the average time cost of main procedures (second). The average number of vertexes and links of latent graphs are 7,000 and 108,360 in the Yelp case, and 34,002 and 418,370 in the Flickr case.

5.7.3 User Study

Our objective is to compare users' performance on conducting the same visual tasks in two views (*view 1: the overall view*; *view 2: the comparison view*). We invited 12 users to attend the study. One of them was an male expert of geographical information systems.

Two female graduate students had knowledge about visual analytics, and the remaining users were undergraduate students (6 males) with diverse majors.

Users were given a standard laptop (1.1GHz Intel Core M3 CPU and 8GB memory) and the system interface was presented in the Google's Chrome browser. Animations in *view 1* were played region by region. The playing time was not recorded. Users could start a replay if needed, and they were allowed to take simple notes during the test.

At the beginning, we gave a brief tutorial on the visual designs and the usage of the system. Then, users conducted the visual tasks listed in Table 5.5. The study was based on experiments of both Flickr and Yelp datasets, and we focused on region B in the Yelp case.

Each user needed to complete 40 tasks (10 for each view, 20 for each dataset). We designed the tasks according to the requirements introduced in Section 5.6.1. For a pair of inverse transformations, we chose one of them as a representative. Therefore, alternative tasks for T_8-T_{10} could be: *How many regions disappear/expand/merge between two time steps?* Considering that the impression on one view will affect the answers of another view, we asked users to finish all tasks for both datasets in *view 1*, and then repeat the procedure in *view 2*. We anticipated that users' performance would be different in two views but similar for two datasets.

Results. We recorded the accuracy and time cost of each task, and the results are presented in Table 5.6. We find that T_1-T_4 have lower accuracy and longer response time in *view 2*, and the same applies to T_6-T_9 in *view 1*. The reason might be that T_1-T_4

require users to have an overall understanding of the evolution. We can observe that *view 1* outperforms *view 2* in these tasks because animations will not be affected by the growth of time. However, in timeline representations, we need to take time to view long-term information by scrolling which causes interruptions in the mental map. T_6-T_9 associate with comparisons of region territories. In *view 1*, some users provided wrong answers as they did comparisons by observation. Others compared the number of hexagonal cells and obtained accurate answers. In *view 2*, tasks can be easily completed by inspecting the width of the stacked rectangles.

Feedback. All users thought the system was helpful. It was easy to use and the two views complemented each other. Some undergraduate students said that, even if they did not have any experience in urban analysis, they could quickly understand what the system presented and start exploring interesting patterns. The graduate students liked the aesthetic representations and the interactive features. They also agreed that links between snapshots were essential as they led the movement of eyes when tracking the evolution of regions. The expert made the following comment: *I can clearly see the difference between region distributions at different time steps, and the categorization of region transformations is especially helpful.* He also pointed out that it took some time to figure out the usage of widgets. We then attached hint texts to these widgets.

T_1	What is the duration of a region?
T_2	What kind of transformations does a region involve?
T_3	Which region experiences only *one-to-one* transformations?
T_4	Which region has its centroids move across the largest area?
T_5	How many regions does the AOI consist of at a time step?
T_6	Which region has the largest territory at a time step?
T_7	Which region has a higher venue density at a time step?
T_8	How many new regions appear at a time step?
T_9	How many regions shrink between two time steps?
T_{10}	How many regions split between two time steps?

Table 5.5: Visual tasks that need to be conducted in both the overall view and the comparison view.

5.8 Discussion and Limitations

The studies have shown that our methods are effective in discovering dynamic regions, and the visual system is helpful in assisting users with exploration tasks. However, there are limitations. First, users need to choose a sufficient k for constructing latent graphs. As we mentioned in Section 5.3.3, a larger k corresponds to less regions with more overlaps. Conversely, we obtain a greater portion of tiny regions (e.g., the area is less than 100 square meters) with a smaller k. To decide on a value, we can progressively increase k until the number of tiny regions (R_t) and the size of the overlapping areas (A_o) are both minimized, but the calculations are time-consuming. Therefore, we provide several k values that have achieved good results in our experiments, and their corresponding R_t and A_o on the current datasets have been calculated. Users can compare them and select the most

	Accuracy				**Time (s)**			
	Flickr		*Yelp*		*Flickr*		*Yelp*	
	view 1	*view 2*	*view 1*	*view 2*	*view 1*	*view 2*	*view 1*	*view 2*
T_1	**100%**	75%	**91.7%**	75%	**5.5**	12.3	**4.8**	10.6
T_2	**83.3%**	75%	**91.7%**	83.3%	**6.1**	13.5	**5.9**	11.2
T_3	**100%**	83.3%	**100%**	66.7%	**4.8**	19.9	**4.7**	20.3
T_4	**91.7%**	41.7%	**83.3%**	33.3%	**5.6**	36.5	**6.4**	39.8
T_5	100%	100%	91.7%	**100%**	8.3	**5.4**	8.8	**5.2**
T_6	50%	**100%**	33.3%	**83.3%**	38.6	**5.5**	43.2	**8.1**
T_7	58.3%	**91.7%**	58.3%	**83.3%**	7.5	**4.6**	8.8	**5.4**
T_8	91.7%	**100%**	100%	100%	6.2	**4.2**	6.3	**4.5**
T_9	41.7%	**91.7%**	41.7%	**100%**	55.7	**4.2**	64.5	**4.1**
T_{10}	100%	100%	91.7%	**100%**	5.8	**4.3**	7.9	**5.7**

Table 5.6: The accuracy and the average time cost of visual tasks. The better performance achieved from the two views is highlighted.

satisfying one. Second, we need to determine the number of topics for the LDA model. The hierarchical Dirichlet process (HDP) [168] is helpful, but it might be infeasible on large corpuses. Similar to the way of k selection, we train models on varying number of topics, and choose the one with the best performance. A model works better if it causes a lower perplexity on a test set of documents. Third, there is no guarantee that venues belonging to the same region are mutually accessible. For the ease of data processing, we use the Eclidean distance to measure the geographical proximity of venues. However, a topological network-based distance, such as the length of the shortest path on a street network, would be a better option.

The reason that we propose a graph-based method rather than using the traditional density-based methods like DBSCAN to cluster venues is two-fold. First, we need to consider two similarity metrics at the same time, one is in geographical space and the other is in semantic space. In the graph-based method, we can conveniently map them to the connectivity of venues and the weight of connections, respectively. Whereas in DBSCAN, we have to design a distance measure that combines the two metrics, like the implementation of ST-DBSCAN (Spatial–Temporal DBSCAN) [169]. Second, DBSCAN requires two parameters, ϵ (the maximum distance between points in a cluster) and $MinPts$ (the minimum number of points required to form a cluster) [170]. It is non-trivial to estimate them. In our method, we only need to determine a k value for constructing graphs, and the frequently adopted values such as 10 and 20 suit most of the scenarios in our tests.

Besides, we can extend our work on revealing evolution patterns by abstracting migration trajectories as attributed graphs. For example, node attributes may include the number of venues, the entropy of topics, and edge attributes can be the distance between region centroids. By applying clustering methods for attributed graphs [171], we are able to find regions that have similar evolution procedures more precisely.

Our method is applicable to the analysis of any spatio-temporal data, e. g., telecommunication records and commuting flows. Such kind of data might be inherently related, and their connection weights should be measured differently, rather than by calculating the semantic similarity of topics. For example, the vertexes of graphs that are constructed from commuting flows can be the departure and destination venues of commuters and edges

can be the routes connecting them. The edge weight denotes the number of trips between two end-venues [84]. Regions that are detected from the graphs are not necessarily areas that provide specific functions, but areas that cover a group of closely related venues and also reflect a high frequency of human activities. Our matching algorithm can expose the popularity of regions at different time and the migration of activity centers.

5.9 Conclusion

We propose a new visual system for analysing the evolution of geographical regions. It consists of two interactive visualization views, one for displaying the animated region migrations and the other for showing the differences and connections between a sequence of snapshots. Compared to conventional methods of urban analysis, we leverage dynamic geo-textual data and update region divisions by employing a graph-based method. Furthermore, the matching algorithm aims to identify different types of region transformations. Users can further explore whether regions have similar evolution patterns. We have conducted two case studies and a user study on real datasets. The results show that the system can effectively assist users in analyzing dynamic features of regions. In the future, we plan to enhance the visual system by automating the selection of parameter values and also apply our methods to exploring more geographical data.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The methods introduced in this thesis aim at achieving the following goals: (1) exposing the structural similarities between graph communities, (2) searching for graph components that have the topological patterns of interest, (3) representing the dynamics of graph properties and (4) improving the usability of the traditional timeline and animation representations.

6.1 Summarizations and Limitations

Chapter 3. We establish a framework for presenting the dynamic relationship between graph communities. Communities can be found by a few state-out-art techniques such as the Louvain algorithm and the Infomap approach. To measure how similar the structures of communities are, we can learn vectorized representations of the corresponding subgraphs by extracting topological features or by Graph Convolutional Networks (GCN). Then the similarity score can be calculated based on the vectors. We use the Multidimensional Scaling technique to decide the spatial proximity of community nodes on the 2D display. For dynamic graphs, a sequence of snapshots are juxtaposed along the timeline.

Consequently, users can easily discern similar communities, and by comparing consecutive snapshots, they can identify communities that involve remarkable changes.

Limitations. We do not consider the attribute of vertices. Hence, vector representations only encode the structural information. If two communities are placed more closely on the display, it only implies that they have similar structures.

Chapter 4. Graph components are classified into topological patterns that users are interested in. Consequently, the diversity of graph structures is reduced. Users can easily locate all components of a specific class. We use animations to show the evolution of graph. The visual stability is improved by generating the layout of components at a global level. Meanwhile, the inner connectivity of components are implied by glyphs that denote the topological patterns. Normally, it is impractical to compare frames as the animation plays. However, we combine snapshots at two consecutive timestamps in one frame, then users can conveniently capture significant graph changes by viewing the transformation of visual encodings.

Limitations. First, the topological patterns only approximate the real structures of graphs. To inspect the details, users still have to expand the collapsed components. Second, the topological patterns have to be determined in advance. Because we have to re-train the classification model after adding or removing patterns. However, changing patterns is totally feasible as long as users do not mind the time cost of training.

Chapter 5. We build a visual system that helps geographical researchers to study the evolution of functional regions. The evolution can be described by a sequence of life states

including the birth, death, expansion and contraction of regions. To delimit functional regions, we construct a dynamic network based on the temporal geo-textual data. Each node of the network represents a venue with attached descriptions of the human activities. Communities detected from the network correspond to regions. Functional regions are then interpreted by the aggregation of activities at all inside venues.

Limitations. To alleviate the problem of data sparsity, we have to fuse the data collected from multiple sources. However, the time cost of pre-processing increases, because we have to manipulate the datasets differently. Currently, the timeline representation is used to show the evolution of regions, but we have to compensate for its poor scalability by adding more interactive features.

6.2 Future Work

6.2.1 Visualizing Multivariate Dynamic Networks

Entities of multivariate networks may possess a large number of attributes which can be intrinsic or induced by topological properties. In social networks, intrinsic attributes could be users' age, gender or the frequency of interaction. Topological attributes include the node centrality, the degree distribution and the clustering coefficient.

Typically, *Parallel Coordinates* (PC) are used to present multivariate data. As shown in Figure 6.1, vertical axes denote attributes. An object is represented by a polygonal line that connects the vertices on axes. Each vertex corresponds to the attribute value of the

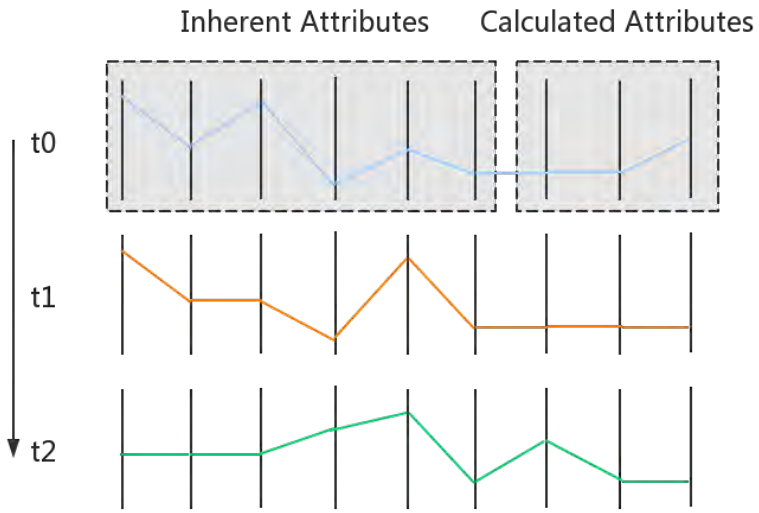


Figure 6.1: Juxtaposed parallel coordinates can potentially be used to represent multivariate dynamic networks.

object. If we further consider the temporal dimension, an intuitive representation is to juxtapose PC for multiple timestamps. But this representation is not scalable along both the horizontal and the vertical directions.

We are planning to use PC-based visualizations to represent the multivariate dynamic networks. Each network node is depicted by a polygonal line, and each axis denotes a variable that is associated with the nodes. The connections between nodes can be demonstrated by linking the corresponding vertices on each vertical axis. Consequently, users can view the density of connections regarding the value distribution of individual variables. Problems that need to be solved include improving the compactness and the scalability of visual representations.

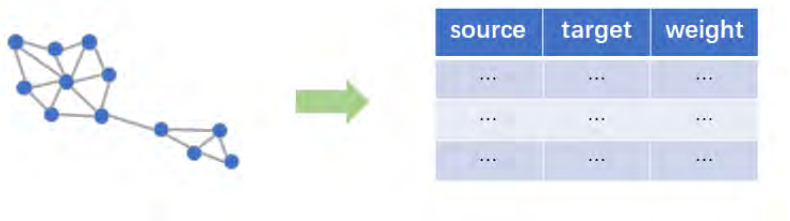


Figure 6.2: Retrieve original data from graph visualizations.

6.2.2 Reverse Engineering

Given static graph drawings without interactive features, users are unable to make changes even if they are not satisfied with the color, shape or layout. Furthermore, it is impossible to automatically obtain the statistical data of the graph.

To add interactive features to static visualizations, the key is to retrieve the original data. Namely, we need to identify vertices and their connectivity. As shown in Figure 6.2, our method takes a graph drawing as the input and the output is a list of connected entities.

Initially, we would like to focus on graph drawings where nodes and links are represented by circles and straight lines, respectively. So, we need to find all graphical elements and their positions. A potential solution is to apply the target detection algorithms that have been well developed in computer vision [172]. But the difficulty substantially increases if there are too many node overlappings and edge crossings in the input.

6.2.3 Story Telling

Many visual systems require users to have the domain knowledge. Otherwise, users have to follow complex instructions to start the exploration. Hence, these systems are not user-friendly to people who are not experts. To let users get a basic understanding of the visualized data, it is necessary to interpret complex visual cues as brief descriptions.

Our objective is to learn stories from the visual system. Stories [173] can be short texts that summarize the statistics of charts and tables. Then, users do not have to know what a symbol denotes in the diagram, such as the quartile segment in box plots. Furthermore, by ranking stories based on their importance, users can quickly capture what they need.

REFERENCES

[1] Robert S Laramee. How to write a visualization research paper: A starting point. In *Computer Graphics Forum*, volume 29, pages 2363–2371. Wiley Online Library, 2010.

[2] Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: A survey. *ACM Comput. Surv.*, 51:35:1–35:37, 2018.

[3] William Thomas Tutte. *Graph theory as I have known it*, volume 11. Clarendon Press, 2012.

[4] George Baciu, Chenhui Li, Yunzhe Wang, and Xiujun Zhang. Cloudets: Cloud-based cognition for large streaming data. In *14th IEEE International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC 2015, Beijing, China, July 6-8, 2015*, pages 333–338, 2015.

[5] George Baciu, Chenhui Li, Yunzhe Wang, and Xiujun Zhang. Cloudet: A cloud-driven visual cognition of large streaming data. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 10(1):12–31, 2016.

[6] George Baciu, Yungzhe Wang, and Chenhui Li. Cognitive visual analytics of multi-dimensional cloud system monitoring data. *International Journal of Software Science and Computational Intelligence (IJSSCI)*, 9(1):20–34, 2017.

[7] Chenhui Li, George Baciu, and Yunzhe Wang. Modulgraph: modularity-based visualization of massive graphs. In *SIGGRAPH Asia 2015 Visualization in High Performance Computing, Kobe, Japan, November 2-6, 2015*, pages 11:1–11:4, 2015.

[8] Chenhui Li, George Baciu, and Yunzhe Wang. Module-based visualization of large-scale graph network data. *Journal of visualization*, 20(2):205–215, 2017.

[9] Yunzhe Wang, George Baciu, and Chenhui Li. Smooth animation of structure evolution in time-varying graphs with pattern matching. In *SIGGRAPH Asia 2017 Symposium on Visualization*, SA '17, pages 12:1–12:8, New York, NY, USA, 2017. ACM.

[10] Yunzhe Wang, George Baciu, and Chenhui Li. Cognitive exploration of regions through analyzing geo-tagged social media data. In *16th IEEE International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC 2017, Oxford, United Kingdom, July 26-28, 2017*, pages 59–64, 2017.

[11] Yunzhe Wang, George Baciu, and Chenhui Li. Cognitive visualization of popular regions discovered from geo-tagged social media data. *IJCINI*, 12(1):14–28, 2018.

[12] Yunzhe Wang, George Baciu, and Chenhui Li. Visualizing functional regions by analysis of geo-textual data. In *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*, pages 25–29. Eurographics Association, 2018.

[13] Warren Sack. Aesthetics of information visualization. *Context Providers: Conditions of Meaning in Media Arts*, pages 123–50, 2011.

[14] Andrea Lau and Andrew Vande Moere. Towards a model of information aesthetics in information visualization. In *2007 11th International Conference Information Visualization (IV'07)*, pages 87–92. IEEE, 2007.

[15] Nick Cawthon and Andrew Vande Moere. The effect of aesthetic on the usability of data visualization. In *2007 11th International Conference Information Visualization (IV'07)*, pages 637–648. IEEE, 2007.

[16] Andrew Vande Moere and Helen Purchase. On the role of design in information visualization. *Information Visualization*, 10(4):356–371, 2011.

[17] Fabian Beck, Michael Burch, and Stephan Diehl. Towards an aesthetic dimensions framework for dynamic graph visualisations. In *2009 13th International Conference Information Visualisation*, pages 592–597. IEEE, 2009.

[18] Mohammad Ghoniem, J-D Fekete, and Philippe Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 17–24. Ieee, 2004.

[19] Helen C Purchase, Eve Hoggan, and Carsten Görg. How important is the "mental map"?–an empirical investigation of a dynamic graph layout algorithm. In *International Symposium on Graph Drawing*, pages 184–195. Springer, 2006.

[20] Chaomei Chen. Top 10 unsolved information visualization problems. *IEEE computer graphics and applications*, 25(4):12–16, 2005.

[21] Chris Bennett, Jody Ryall, Leo Spalteholz, and Amy Gooch. The aesthetics of graph visualization. *Computational aesthetics*, 2007:57–64, 2007.

[22] Helen Gibson, Joe Faith, and Paul Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information visualization*, 12(3-4):324–357, 2013.

[23] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[24] Stephen G. Kobourov. Spring embedders and force directed graph drawing algorithms. *CoRR*, abs/1201.3011, 2012.

[25] Michael J Bannister, David Eppstein, Michael T Goodrich, and Lowell Trott. Force-directed graph drawing using social gravity and scaling. In *International Symposium on Graph Drawing*, pages 414–425. Springer, 2012.

[26] P. EADES. A heuristics for graph drawing. *Congressus Numerantium*, 42:146–160, 1984.

[27] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.

[28] Arne Frick, Andreas Ludwig, and Heiko Mehldau. A fast adaptive layout algorithm for undirected graphs (extended abstract and system demonstration). In *International Symposium on Graph Drawing*, pages 388–403. Springer, 1994.

[29] Bruce Hendrickson and Robert W Leland. A multi-level algorithm for partitioning graphs. *SC*, 95:28, 1995.

[30] Daniel Archambault, Tamara Munzner, and David Auber. Topolayout: Multilevel graph layout by topological features. *IEEE transactions on visualization and computer graphics*, 13(2):305–317, 2007.

[31] Chris Muelder and Kwan-Liu Ma. Rapid graph layout using space filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 14(6), 2008.

[32] Anastasia Bezerianos, Fanny Chevalier, Pierre Dragicevic, Niklas Elmqvist, and Jean-Daniel Fekete. Graphdice: A system for exploring multivariate social networks. In *Computer Graphics Forum*, volume 29, pages 863–872. Wiley Online Library, 2010.

[33] Hiroshi Hosobe. A high-dimensional approach to interactive graph visualization. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1253–1257. ACM, 2004.

[34] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A taxonomy and survey of dynamic graph visualization. In *Computer Graphics Forum*. Wiley Online Library, 2016.

[35] Kazuo Misue, Peter Eades, Wei Lai, and Kozo Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183 – 210, 1995.

[36] Yaniv Frishman and Ayellet Tal. Dynamic drawing of clustered graphs. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 191–198. IEEE, 2004.

[37] Thomas E Gorochowski, Mario di Bernardo, and Claire S Grierson. Using aging to visually uncover evolutionary processes on networks. *IEEE transactions on visualization and computer graphics*, 18(8):1343–1352, 2012.

[38] Yi-Yi Lee, Chun-Cheng Lin, and Hsu-Chun Yen. Mental map preserving graph drawing using simulated annealing. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation-Volume 60*, pages 179–188. Australian Computer Society, Inc., 2006.

[39] Kun-Chuan Feng, Chaoli Wang, Han-Wei Shen, and Tong-Yee Lee. Coherent time-varying graph drawing with multifocus+ context interaction. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1330–1342, 2012.

[40] Stephan Diehl, Carsten Görg, and Andreas Kerren. Preserving the mental map using foresighted layout. In *Data Visualization 2001*, pages 175–184. Springer, 2001.

[41] Michael Burch, Corinna Vehlow, Fabian Beck, Stephan Diehl, and Daniel Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE*

Transactions on Visualization and Computer Graphics, 17(12):2344–2353, 2011.

[42] Martin Greilich, Michael Burch, and Stephan Diehl. Visualizing the evolution of compound digraphs with timearctrees. In *Computer Graphics Forum*, volume 28, pages 975–982. Wiley Online Library, 2009.

[43] Matthew Brehmer, Bongshin Lee, Benjamin Bach, Nathalie Henry Riche, and Tamara Munzner. Timelines revisited: A design space and considerations for expressive storytelling. *IEEE Transactions on Visualization and Computer Graphics*, 2016.

[44] `http://www.infowetrust.com/creative-routines/`.

[45] Marc Weber, Marc Alexa, and Wolfgang Müller. Visualizing time-series on spirals. In *Infovis*, volume 1, pages 7–14, 2001.

[46] Nbalockout. `http://visual.ly/nba-lockout-1`.

[47] Datelens. `http://www.cs.umd.edu/hcil/datelens/`.

[48] Benjamin Bach, Conglei Shi, Nicolas Heulot, Tara Madhyastha, Tom Grabowski, and Pierre Dragicevic. Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE transactions on visualization and computer graphics*, 22(1):559–568, 2016.

[49] Milos Krstajic, Enrico Bertini, and Daniel Keim. Cloudlines: Compact display of event episodes in multiple time-series. *IEEE transactions on visualization and*

computer graphics, 17(12):2432–2439, 2011.

[50] Saturnino Luz and Masood Masoodian. Visualisation of parallel data streams with temporal mosaics. In *Information Visualization, 2007. IV'07. 11th International Conference*, pages 197–202. IEEE, 2007.

[51] Benjamin B Bederson, Aaron Clamage, Mary P Czerwinski, and George G Robertson. Datelens: A fisheye calendar interface for pdas. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 11(1):90–119, 2004.

[52] Histography. `http://histography.io/`.

[53] Ilya Boyandin, Enrico Bertini, and Denis Lalanne. A qualitative study on the exploration of temporal changes in flow maps with animation and small-multiples. In *Computer Graphics Forum*, volume 31, pages 1005–1014. Wiley Online Library, 2012.

[54] Steffen Hadlak, Hans-Jorg Schulz, and Heidrun Schumann. In situ exploration of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2334–2343, 2011.

[55] Sébastien Rufiange and Michael J McGuffin. Diffani: Visualizing dynamic graphs with a hybrid of difference maps and animation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2556–2565, 2013.

[56] Fabian Beck, Michael Burch, Corinna Vehlow, Stephan Diehl, and Daniel Weiskopf. Rapid serial visual presentation in dynamic graph visualization. In *2012*

IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pages 185–192. IEEE, 2012.

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

[58] Ji Soo Yi, Youn ah Kang, John Stasko, and Julie Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, 13(6):1224–1231, 2007.

[59] Daniel F Keefe and Tobias Isenberg. Reimagining the scientific visualization interaction paradigm. *IEEE COMPUTER*, 46(5):51–57, 2013.

[60] Yifan Hu, Lei Shi, Farid Aadil, Latif Izdihar, and Carol A. Fernando. Visualizing large graphs. 2015.

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

[62] Quan Hoang Nguyen, Seok-Hee Hong, Peter Eades, and Amyra Meidiana. Proxy graph: Visual quality metrics of big graph sampling. *IEEE Transactions on Visualization and Computer Graphics*, 23(6):1600–1611, 2017.

[63] Yanhong Wu, Nan Cao, Daniel Archambault, Qiaomu Shen, Huamin Qu, and Wei-

wei Cui. Evaluation of graph sampling: A visualization perspective. *IEEE transactions on visualization and computer graphics*, 23(1):401–410, 2017.

[64] Christos Gkantsidis, Milena Mihail, and Ellen Zegura. Spectral analysis of internet topologies. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 1, pages 364–374. IEEE, 2003.

[65] Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.

[66] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.

[67] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[68] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[69] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[70] Emile Aarts and Jan Korst. Simulated annealing and boltzmann machines. 1988.

[71] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[72] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys (csur)*, 45(4):43, 2013.

[73] Cfinder:. `http://www.cfinder.org/`, November 2016.

[74] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[75] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[76] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, 2010.

[77] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[78] Stéphan Clémençon, Hector De Arazoza, Fabrice Rossi, and Viet Chi Tran. Hierarchical clustering for graph visualization. *arXiv preprint arXiv:1210.5693*, 2012.

[79] Hong Zhou, Panpan Xu, Xiaoru Yuan, and Huamin Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013.

[80] Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on visualization and computer graphics*, 12(5):741–748, 2006.

[81] Huamin Qu, Hong Zhou, and Yingcai Wu. Controllable and progressive edge clustering for large networks. In *International Symposium on Graph Drawing*, pages 399–404. Springer, 2006.

[82] Danny Holten and Jarke J Van Wijk. Force-directed edge bundling for graph visualization. In *Computer graphics forum*, volume 28, pages 983–990. Wiley Online Library, 2009.

[83] Emden R Gansner and Yehuda Koren. Improved circular layouts. In *International Symposium on Graph Drawing*, pages 386–398. Springer, 2006.

[84] Tatiana Von Landesberger, Felix Brodkorb, Philipp Roskosch, Natalia Andrienko, Gennady Andrienko, and Andreas Kerren. Mobilitygraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE transactions on visualization and computer graphics*, 22(1):11–20, 2016.

[85] Tim Dwyer, Christopher Mears, Kerri Morgan, Todd Niven, Kim Marriott, and Mark Wallace. Improved optimal and approximate power graph compression for

clearer visualisation of dense graphs. In *2014 IEEE Pacific Visualization Symposium*, pages 105–112. IEEE, 2014.

[86] Cody Dunne and Ben Shneiderman. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3247–3256. ACM, 2013.

[87] Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: a survey. *ACM Computing Surveys (CSUR)*, 51(2):35, 2018.

[88] Jiaxing Shang, Lianchen Liu, Feng Xie, Zhen Chen, Jiajia Miao, Xuelin Fang, and Cheng Wu. A real-time detecting algorithm for tracking community structure of dynamic networks. *CoRR*, abs/1407.2683, 2012.

[89] Francesco Folino and Clara Pizzuti. An evolutionary multiobjective approach for community discovery in dynamic networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1838–1852, 2014.

[90] Giulio Rossetti, Luca Pappalardo, Dino Pedreschi, and Fosca Giannotti. Tiles: an online algorithm for community discovery in dynamic social networks. *Machine Learning*, 106(8):1213–1241, 2017.

[91] Laetitia Gauvin, André Panisson, and Ciro Cattuto. Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *PloS one*, 9(1):e86028, 2014.

[92] Daniel Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon. Visual analytics: Definition, process, and challenges. In *Information visualization*, pages 154–175. Springer, 2008.

[93] Kathleen M Carley. *Dynamic network analysis*. Citeseer, 2003.

[94] Jarke J Van Wijk. The value of visualization. In *VIS 05. IEEE Visualization, 2005.*, pages 79–86. IEEE, 2005.

[95] Jae-wook Ahn, Catherine Plaisant, and Ben Shneiderman. A task taxonomy for network evolution analysis. *IEEE transactions on visualization and computer graphics*, 20(3):365–376, 2014.

[96] Chenhui Li, George Baciu, and Yunzhe Wang. Module-based visualization of large-scale graph network data. *Journal of Visualization*, pages 1–11, 2016.

[97] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–5. ACM, 2006.

[98] James Moody, Daniel McFarland, and Skye Bender-deMoll. Dynamic network visualization1. *American journal of sociology*, 110(4):1206–1241, 2005.

[99] James Abello, Tina Eliassi-Rad, and Nishchal Devanur. Detecting novel discrepancies in communication networks. In *2010 IEEE International Conference on Data Mining*, pages 8–17. IEEE, 2010.

[100] Bernard Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, 2000.

[101] Stephen J Read, Eric J Vanman, and Lynn C Miller. Connectionism, parallel constraint satisfaction processes, and gestalt principles:(re) introducing cognitive dynamics to social psychology. *Personality and Social Psychology Review*, 1(1):26–53, 1997.

[102] Keith V Nesbitt and Carsten Friedrich. Applying gestalt principles to animated visualizations of network data. In *Information Visualisation, 2002. Proceedings. Sixth International Conference on*, pages 737–743. IEEE, 2002.

[103] Benjamin Bach, Emmanuel Pietriga, and Jean-Daniel Fekete. Graphdiaries: animated transitions andtemporal navigation for dynamic networks. *IEEE transactions on visualization and computer graphics*, 20(5):740–754, 2014.

[104] Krist Wongsuphasawat and Ben Shneiderman. Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pages 27–34. IEEE, 2009.

[105] Khairi Reda, Chayant Tantipathananandh, Andrew Johnson, Jason Leigh, and Tanya Berger-Wolf. Visualizing the evolution of community structures in dynamic social networks. In *Computer Graphics Forum*, volume 30, pages 1061–1070. Wiley Online Library, 2011.

[106] Corinna Vehlow, Fabian Beck, Patrick Auwärter, and Daniel Weiskopf. Visualizing the evolution of communities in dynamic graphs. In *Computer Graphics Forum*, volume 34, pages 277–288. Wiley Online Library, 2015.

[107] Yanhong Wu, Naveen Pitipornvivat, Jian Zhao, Sixiao Yang, Guowei Huang, and Huamin Qu. egoslider: Visual analysis of egocentric network evolution. *IEEE transactions on visualization and computer graphics*, 22(1):260–269, 2016.

[108] Yu-Ru Lin, Jimeng Sun, Nan Cao, and Shixia Liu. Contextour: Contextual contour visual analysis on dynamic multi-relational clustering. In *SIAM Data Mining Conference (accepted)*. SIAM, 2010.

[109] Qingsong Liu, Yifan Hu, Lei Shi, Xinzhu Mu, Yutao Zhang, and Jie Tang. Egonetcloud: Event-based egocentric dynamic network visualization. In *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*, pages 65–72. IEEE, 2015.

[110] Jean-Daniel Fekete, Jarke J Van Wijk, John T Stasko, and Chris North. The value of information visualization. In *Information visualization*, pages 1–18. Springer, 2008.

[111] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116. ACM, 2004.

[112] James Abello, Frank Van Ham, and Neeraj Krishnan. Ask-graphview: A large

scale graph visualization system. *IEEE transactions on visualization and computer graphics*, 12(5):669–676, 2006.

[113] Emden R Gansner, Yehuda Koren, and Stephen C North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):457–468, 2005.

[114] Junzo Kamahara, Tomofumi Asakawa, Shinji Shimojo, and Hideo Miyahara. A community-based recommendation system to reveal unexpected interests. In *11th international multimedia modelling conference*, pages 433–438. IEEE, 2005.

[115] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98, 2008.

[116] M Mitrović, Georgios Paltoglou, and B Tadić. Networks and emotion-driven user communities at popular blogs. *The European Physical Journal B*, 77(4):597–609, 2010.

[117] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[118] Gennaro Cordasco and Luisa Gargano. Community detection via semi-synchronous label propagation algorithms. In *2010 IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA)*, pages 1–8. IEEE, 2010.

[119] Leonid Barenboim and Michael Elkin. Distributed (δ+ 1)-coloring in linear (in δ) time. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 111–120. ACM, 2009.

[120] Zhao Yang, René Algesheimer, and Claudio J Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, 6:30750, 2016.

[121] Oh-Hyun Kwon, Tarik Crnovrsanin, and Kwan-Liu Ma. What would a graph look like in this layout? a machine learning approach to large graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24:478–488, 2018.

[122] Mustafa Hajij, Bei Wang, Carlos Scheidegger, and Paul Rosen. Visual detection of structural changes in time-varying graphs using persistent homology. In *Pacific Visualization Symposium (PacificVis), 2018 IEEE*, pages 125–134. IEEE, 2018.

[123] Michele Berlingerio, Danai Koutra, Tina Eliassirad, and Christos Faloutsos. Netsimile: A scalable approach to size-independent network similarity. *Computer Science*, 12(1):28(1–28), 2012.

[124] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[125] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[126] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[127] Openstack. `https://www.openstack.org/`.

[128] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[129] Snap. `https://snap.stanford.edu/index.html`.

[130] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.

[131] Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[132] Mark Harrower and Cynthia A Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[133] Mansoureh Takaffoli, Reihaneh Rabbany, and Osmar R. Zaïane. Community evolution prediction in dynamic social networks. *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 9–16, 2014.

[134] Giulio Rossetti, Luca Pappalardo, Dino Pedreschi, and Fosca Giannotti. Tiles: an online algorithm for community discovery in dynamic social networks. *Machine*

Learning, 106:1213–1241, 2016.

[135] Chenhui Li, George Baciu, and Yunzhe Wang. Modulgraph: Modularity-based visualization of massive graphs. In *SIGGRAPH Asia 2015 Visualization in High Performance Computing*, SA '15, pages 11:1–11:4, 2015.

[136] Tim Dwyer. Scalable, versatile and simple constrained graph layout. In *Computer Graphics Forum*, volume 28, pages 991–998. Wiley Online Library, 2009.

[137] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013.

[138] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[139] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[140] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[141] Paolo Federico, Jurgen Pfeffer, Wolfgang Aigner, Silvia Miksch, and Lukas Zenk. Visual analysis of dynamic networks using change centrality. In *Proceedings of*

the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), pages 179–183. IEEE Computer Society, 2012.

[142] Jure Leskovec, Daniel P Huttenlocher, and Jon M Kleinberg. Governance in social media: A case study of the wikipedia promotion process. In *ICWSM*, pages 98–105, 2010.

[143] G Kossinets. Processed wikipedia edit history, 2012.

[144] George Karypis and Vipin Kumar. Metis–unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[145] Gang Pan, Guande Qi, Zhaohui Wu, Daqing Zhang, and Shijian Li. Land-use classification using taxi gps traces. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):113–123, 2013.

[146] Dongyu Liu, Di Weng, Yuhong Li, Jie Bao, Yu Zheng, Huamin Qu, and Yingcai Wu. Smartadp: Visual analytics of large-scale taxi trajectories for selecting billboard locations. *IEEE transactions on visualization and computer graphics*, 23(1):1–10, 2016.

[147] Alan M MacEachren, Anuj Jaiswal, Anthony C Robinson, Scott Pezanowski, Alexander Savelyev, Prasenjit Mitra, Xiao Zhang, and Justine Blanford. Senseplace2: Geotwitter analytics support for situational awareness. In *Visual analytics science and technology (VAST), 2011 IEEE conference on*, pages 181–190. IEEE, 2011.

[148] Nicholas Jing Yuan, Yu Zheng, Xing Xie, Yingzi Wang, Kai Zheng, and Hui Xiong. Discovering urban functional zones using latent activity trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):712–725, 2015.

[149] Wenchao Wu, Yixian Zheng, Nan Cao, Haipeng Zeng, Bing Ni, Huamin Qu, and Lionel M Ni. Mobiseg: Interactive region segmentation using heterogeneous mobility data. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pages 91–100. IEEE, 2017.

[150] Carlo Ratti, Stanislav Sobolevsky, Francesco Calabrese, Clio Andris, Jonathan Reades, Mauro Martino, Rob Claxton, and Steven H Strogatz. Redrawing the map of great britain from a network of human interactions. *PloS one*, 5(12):e14248, 2010.

[151] Justin Cranshaw, Raz Schwartz, Jason I. Hong, and Norman M. Sadeh. The livehoods project: Utilizing social media to understand the dynamics of a city. In *ICWSM*, 2012.

[152] Adel Hlaoui and Shengrui Wang. A direct approach to graph clustering. *Neural Networks and Computational Intelligence*, 4(8):158–163, 2004.

[153] Matthew J Rattigan, Marc Maier, and David Jensen. Graph clustering with network structure indices. In *Proceedings of the 24th international conference on Machine learning*, pages 783–790. ACM, 2007.

[154] Andrea Capocci, Vito DP Servedio, Guido Caldarelli, and Francesca Colaiori. Detecting communities in large networks. *Physica A: Statistical Mechanics and its Applications*, 352(2-4):669–676, 2005.

[155] Peter Ronhovde and Zohar Nussinov. Multiresolution community detection for megascale networks by information-based replica correlations. *Physical Review E*, 80(1):016109, 2009.

[156] Kevin Weiler and Peter R. Atherton. Hidden surface removal using polygon area sorting. In *SIGGRAPH*, 1977.

[157] Scott D ROTh. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

[158] Gabor J Szekely and Maria L Rizzo. Hierarchical clustering via joint between-within distances: Extending ward's minimum variance method. *Journal of classification*, 22(2):151–183, 2005.

[159] Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Netsimile: A scalable approach to size-independent network similarity. *CoRR*, abs/1209.2684, 2012.

[160] Natalia Andrienko, Gennady Andrienko, and Peter Gatalsky. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, 14(6):503–541, 2003.

[161] C. Li, G. Baciu, and Y. Han. Streammap: Smooth dynamic visualization of high-density streaming points. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1381–1393, March 2018.

[162] Graham McNeill and Scott A Hale. Generating tile maps. In *Computer Graphics Forum*, volume 36, pages 435–445. Wiley Online Library, 2017.

[163] Rafael G Cano, Kevin Buchin, Thom Castermans, Astrid Pieterse, Willem Sonke, and Bettina Speckmann. Mosaic drawings and cartograms. In *Computer Graphics Forum*, volume 34, pages 361–370. Wiley Online Library, 2015.

[164] Chris Wylie, Gordon Romney, David Evans, and Alan Erdahl. Half-tone perspective drawings by computer. In *Proceedings of the November 14-16, 1967, fall joint computer conference*, pages 49–58. ACM, 1967.

[165] Siming Chen, Shuai Chen, Zhenhuang Wang, Jie Liang, Xiaoru Yuan, Nan Cao, and Yadong Wu. D-map: Visual analysis of ego-centric information diffusion patterns in social media. In *Visual Analytics Science and Technology (VAST), 2016 IEEE Conference on*, pages 41–50. IEEE, 2016.

[166] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[167] Yelp dataset challenge. `https://www.yelp.com/dataset/challenge`, 2017.

[168] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[169] Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.

[170] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[171] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, pages 505–516. ACM, 2012.

[172] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[173] Robert Kosara and Jock Mackinlay. Storytelling: The next step for visualization. *Computer*, 46(5):44–50, 2013.