博士論文

Visualizing Dynamic Networks using Matrix Technology

（行列技術を用いた動的ネットワーク可視化）

斉　済

Abstract

This dissertation makes contributions to the field of dynamic network visualization related to knowledge discovery and analysis of the evolution of community structure over time. Our approach provide a framework involving three levels of dynamic network visualization as a circle of data analysis: underlying algorithms processing large-scale data streams as the lowest level, appropriately designed visual interface including all the human-computer interactions as the middle level, and exploring method with pattern of potential discoveries as the highest level. Experiments are conducted for evaluating the approach from different angles.

In this work, I present both the two underlying algorithms for data processing. One is the dynamic matrix seriation algorithm for arranging the visualization layout over time. Comparing with previous outstanding algorithms in this field, our approach makes consideration of both the smoothness of matrix permutation and the highlight of hidden patterns at the same time. Another algorithm is for detecting the evolution of network community structure over time, where the main idea of the algorithm is dynamically capturing the splitting/merging events of communities. The performance of our algorithm is proved to be more stable when networks are more complex with vague boundaries between communities. Meanwhile, I also present a 2 dimensional, matrix-like, and animation-based design of visualization, where each decision of feature selection is for the purpose of reducing the interface cost of computation and the learning cost of users. Finally, I present a pattern of potential meaningful knowledge on the visualization, which is inspired by the process of chance discovery via KeyGraph.

Together, this visualization approach suggests a way of knowledge discovery over rapid data stream of networks, and we suggest the application of our approach is in the domain of real-time analytics on big data. Potential application scenarios on real world tasks and future research directions are discussed.

Acknowledgements

First, I wish to thank my supervisor and mentor, Professor Yukio Ohsawa, for opening the bright window of human-computer interaction for me. He helps me to find out a new world of research combining machine learning and data analysis with human subjects and actions. I sincerely appreciate his trust, patience, and ongoing support.

Second, I want to thank my girlfriend, Shihan Wang, for her wholehearted support in my daily life and research. I can always be inspired by her magic whimsy. Also, I thank for my family for supporting me unconditionally for everything. Wherever I am living and whatever I am facing, there is a place I believe I can always go back anytime.

Finally, I thank all the colleges and friends in Ohsawa Lab and the University of Tokyo for guiding me and helping me over the years. Also, I thank the Japan Science and Technology Agency (JST) for supporting part of this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Network Dynamics

Networks in the real world involve dynamics generally. The network dynamics can be defined as the changes in network topology over time. By taking about the changes on network topology, there exist two perspectives: nodes may come and go, while links may recover and pass away. The duration of network dynamics is always transient, which means the topology changes on networks always occur for a short period only. Also, network dynamics are constantly occurring over time, but the frequency may vary. In general, this pace is due to the real scenario of application. For example, Figure 1.1 presents a comparison of the Twitter network before and after the 2011 Japanese earthquake. It is clearly seen that the communities within the network growth much larger and denser after the disaster than before, while the top hot trends also changed as the scale of the node color shows.

There are many illustrative problems worked on in the domain of dynamic network analysis, such as learning and extracting theories of network evolution, decay, and adoption; developing control process of dynamic networks online; forecasting changes in existing networks; tracking groups in networks overtimes; etc. Definitely, it also involves developing techniques to visualize network dynamics overall or locally.

In this dissertation, the network dynamic we would like to describe through

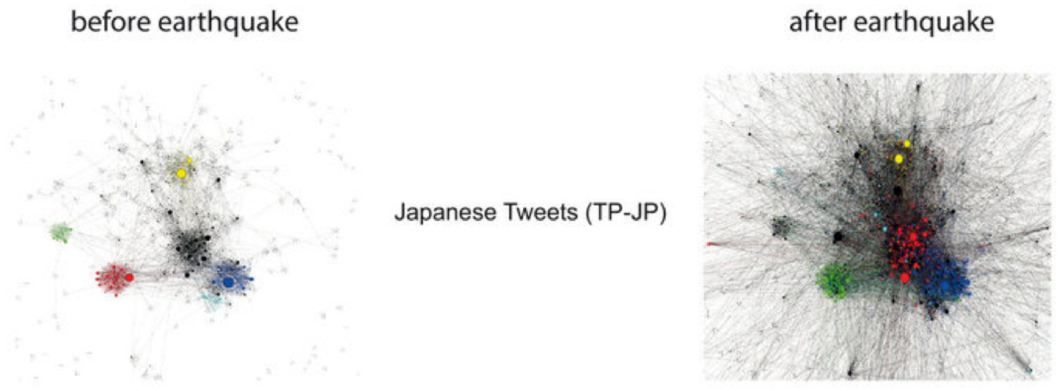

Figure 1.1: Visualization of the Twitter network before and after the 2011 Japanese earthquake, where different color shows different shows the communities with different topics of chatting [1].

information visualization is the evolution of network community structure over time. There has been previous research stating that the local community structure of networks, especially large-scale networks in the real world, is general and has been paid much attention in the dynamic network analysis field [2]. Also, we considered that it is difficult and less meaningful to observe the topological changes on the level of individual nodes and links in case of visualizing large-scale networks.

Meanwhile, the changes in network topology is defined as only the increase of nodes/links, since the networks we considered in this dissertation are incremental networks such as traffic networks, chatting networks on social media, neural networks of human brain, etc., where nodes and links will not disappear once added into the network. Also, the input of the network is a continuous sequence of links instead of a stream from multiple distributed sources, as we assume there is only one link coming to the network every time a topological change occurs.

1.2 Problem Statement

As a powerful and essential means, visualization technology assists analyzing and understanding the evolution of dynamic networks over time. Visualizing dynamic networks face several challenges. First, the information overload is a general issue caused by too complicated user interfaces. Second, huge interface cost of computation is another problem in this domain. This is correlated to the improving requirement of analysis on a large dataset, especially for the case of stream data mining, where data keep coming rapidly in the format of a data stream in real-time. Also, visualization should be combined with reasonable exploring methods for making meaningful observations from the networks.

The aim of this dissertation is to introduce an online visualization technology that efficiently represents the evolution process of large dynamic networks in real-time. As an assistant technology, data visualization should be perfectly embedded into the procedure of data analysis as an important circle. Based on this idea, we design our visualization to involve three levels of implementations:

- At the lowest level, the underlying algorithms dynamically arrange the layout of visualization and detect the evolution of network structure for supporting the online visualization. This level is the entrance of input data and the foundation of the visualization.

- At the middle level, the visual interface conducts all the interactions between human and machine. This part is the core of visualization technology because all the insights of data from users are generated based on this part.

- At the highest level, an exploring method combining with the visualization is introduced for making observations that have the potential to be meaningful. Through this level, data are indeed transferred into knowledge that can support further analysis and decision making.

Figure 1.2 provides an intuitive explanation of the role of visualization technology and the functions of each level of design. Together, work in this dissertation is designed to assist the understanding and discovering of dynamic networks in real time.

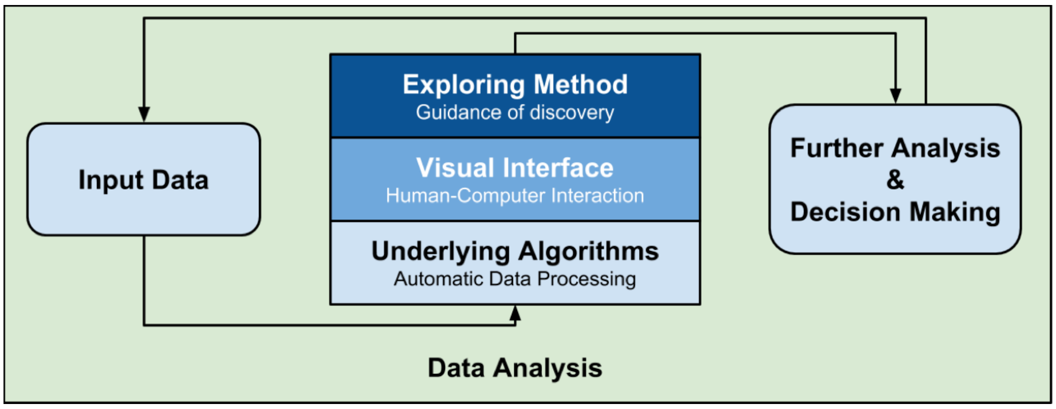

Figure 1.2: Sketch map of the procedure of data analysis and the role of data visualization technology

1.3 Background and Motivation

Dynamic networks have been employed in many important aspects of our life, such as social media sites (SNS), traffic dispatching, stock marketing, etc. One of the most valuable approaches to understanding dynamic networks and especially making complicated network dataset more approachable to most people is data visualization technology. Appropriate visualization helps improve the efficiency of knowledge mining on networks. Generally, nearly all the networks in the real world are dynamic, where both content and structure of networks evolve over time passage. So many network evolutions are going on around us in daily life. For instance, our everyday chatting on SNS builds the dynamic network of our daily social relationship; the transportation in urban area itself is a big dynamic network of traffic stream; electronic signals are transmitted on the neural network in human brains every moment. Researchers in domains like real-time analytics believe that tracking and reasoning the evolution of networks is meaningful for decision making. In this background, online visualization of dynamic networks makes it possible to observe the network evolution and make discoveries in real time. As a result, research on dynamic network visualization attracts much attention from researchers of data visualization. By reviewing the research in past decades, several challenges still remain in this domain.

Network dynamics as additional complexity leads to information overload

For visualizing dynamic networks, especially when networks get larger, visual interface tend to be more complicated with too many parameters to be adjusted and views to be shown in a single window [3]. When this complexity rises above a certain level, users may feel too difficult to learn the visualization, probably as well as the network data. This issue is summarized as information overload, which is significant on tasks of dynamic network visualization and navigation [4]. To avoid information overload, it should be considered by visualization developers that which feature(s) should be highlighted within the visual interface, so that users can quickly focus on the discovering and analyzing process.

Figure 1.3 shows an example of a complicated user interface belonging to a visual software for data analysis. This interface involves 16 subviews, where 8 of them should be operated by users at the same time. With this complicated UI, it is difficult to make a move in a short time even for trained users. And this complexity definitely increases the learning cost of users.

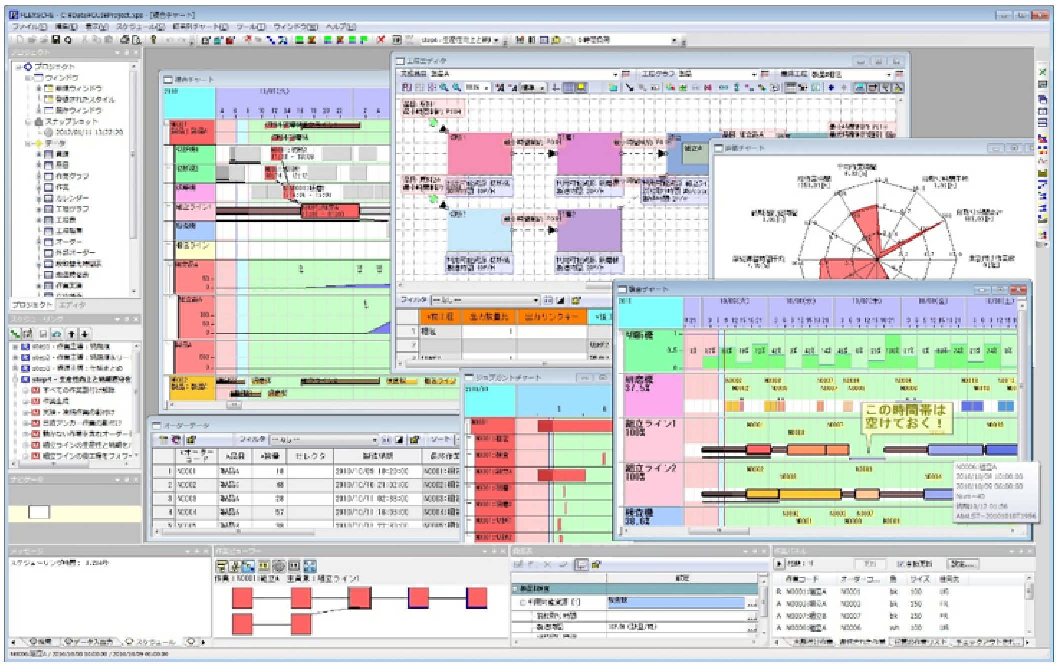

Figure 1.3: A sample interface of a visualization software

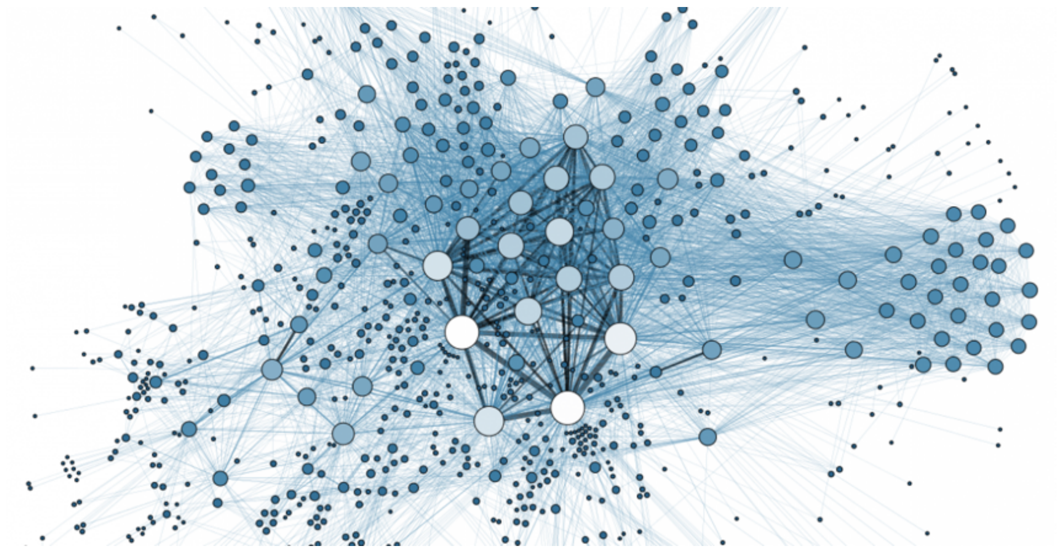

Figure 1.4: Example presentation of a large network represented by D3.js

Visualizing large dynamic network in real time is challenging

When network size gets large enough, the computational expense can become very high. Here, two partitions of computation are involved. One is the execution of the underlying algorithms. There exists computation between raw data streams and structured data for supporting visualization. Generally, this work is automatically completed by computers deal to its large-scaling. Another is the interface cost, which corresponds to essential drawing and animation of shapes, colors, opacity, and any other features for the purpose of visualization. For example, if visualizing a network with 5000 nodes and 100000 links using a node-link graph, there will be at least $5000 \times 2 + 100000 = 110000$ entities displayed at the same time, where $\times 2$ corresponds to the 5000 labels of nodes. In this case, the computational cost of the underlying algorithms is the cost of dynamically arranging the network layout, and the cost of the visual interface comes from maintaining and animating all the 110000 entities based on the layout arrangement. Figure 1.4 shows a capture of a large network represented using D3.js. Definitely, this high cost cannot be ignored in online visualization tasks.

Methods should be declared to guide users to explore and understand the data

Generally, previous methods only focus on the visualization task itself but ignore the role that visualization technology play in the whole procedure of data analysis. However, even with an appropriate visualization, it can still be difficult to make a meaningful observation without any guidance, especially when the data keeps varying. For this reason, the exploring method for visualization is also essential, where guidance is for suggesting users about what kind of observation is possible to be meaningful, so that users can quickly put their energy into the discovering process, instead of wasting time on blind navigation.

1.4 Summary of Contributions

As we mentioned in the previous section, this dissertation presents an online visualization of dynamic networks based on three levels: (1) underlying algorithms dynamically arranging the visual layout, (2) visual interface aiming at reducing both the interface cost of computation and the learning cost of users, and (3) exploring method for visualization focusing on guiding the process of knowledge discovery. This section briefly illustrates the main ideas of the three levels of visualization in this dissertation and outlines the contributions discussed in later chapters.

1.4.1 Contributions in Visual Interface

Visualization in this work is designed to be a 2-dimensional, matrix-like, and animation-based approach. The major presentation form is the adjacency matrix of communities in networks, where major information of network community structure is remained by involving less visual entities comparing with employing adjacency matrix of nodes in the traditional way. On the other hand, 2-dimensional representation speeds up the generation and variation of visualization, while the animation-based representation of time dimension simplifies the recognizing and tracking of network structures. The core idea of this design is for reducing the interface cost of computation and the recognizing cost of users.

This work is described in Chapter 2. Preliminary study of this work is published in [5], and a follow-up study is published in [6]

Contributions

- a novel design of dynamic network visualization that reduces both the recognizing cost of network structure and the computational cost of visual interface;
- case study shows how the visualization assist users in knowledge discovering task on networks, against both benchmark networks and real datasets.

1.4.2 Contributions in Underlying Algorithms

For dynamically detecting the network community structure over time passage, an incremental algorithm of community detection is developed. Communities are split or merged based on the splitting/merging condition, which is defined as community bipartition ratio. Experimental evaluation shows statistical evidence of a more stable performance on networks with different degrees of network complexity, comparing with a number of previous algorithms widely used.

Another algorithm implemented in this work is an incremental algorithm of matrix seriation (matrix reordering in other words), which is for arranging the layout of visualization. Hidden knowledge can be highlighted through this arrangement for improving the efficiency of knowledge discovery. Comparing with previous methods optimizing different coefficients, our algorithm balances the smoothness of matrix permutation and the highlighting presentation of hidden information.

Both of the two algorithms are compared with previous approaches against benchmark datasets for performance evaluation. This work is described in

Chapter 3 with more details. A preliminary study of the matrix seriation algorithm is published in [5], and a follow-up study is going to be published in [7]. The community detection algorithm is published in [6].

Contributions

- a novel incremental algorithm of network community detection, which provides a stable performance on networks with different degree of complexity.
- a novel incremental algorithm of matrix seriation, which balance smoothing display and highlighting features.
- Statistical and empirical evaluation of the two algorithms against both benchmark networks and real datasets.

1.4.3 Contributions in Exploring Method

Inspired by the process of chance discovery via KeyGraph [8], a pattern of potential meaningful discovery is defined for guiding the navigation process on the visualization. By following the guidance, the knowledge discovery process by users is expected to be more intuitive and simple. A case study against a history logging dataset of book co-purchasing on Amazon reveals empirical evidence of efficient discovery following the guidance. This work is illustrated in Chapter 4 with related experiments in Chapter 5.

Contributions

- a detailed exploring method for discovering rare but meaningful knowledge behind network data, combining with visualization technology.

1.5 Potential Impact and Research Directions

Originally, data analysis and knowledge discovery on a static database is challenging, while problem tends to be more complex in the case of dynamic data streams. Large-scaling data streams bring difficulty of analytics in real time, and the uncertainty of online data streams increases the troubles. Contributions of this dissertation have the potential to assist real-time analytics on dynamic networks in domains such as social medias, e-commerce, transportation, and all the other domains building on dynamic networks and can benefit from real-time computing and analytics.

Application scenarios of our visualization are expected to be as follows for example:

- Instead of analyzing user relationships and behaviors on a long-term accumulated data, it is possible to successfully build new relationships between social media users in different communities of interest, only based on the network behaviors in real time.
- A real-time analysis of customer-store networks for online B2C e-commercial sites could be useful, since an appropriate warehouse management always brings revenue growth, especially during big sales.

More detailed discussion about the potential impact and research directions is in Chapter 6. In summary, we see the opportunities to assist real-time analytics based on dynamic networks.

1.6 Roadmap

The remaining chapters of this dissertation is organized as follows: detailed design and implementation of the visual interface is introduced in Chapter 2; the two underlying algorithms are illustrated in Chapter 3; the exploring method for this visualization is presented in Chapter 4; all the experimental evaluation works are included in Chapter 5; a summarization of this dissertation with further research directions is presented in Chapter 6. Detailed structure within each chapter is described as follows:

Chapter 2 begins by introducing the design of visual interface based on three main features of visual design: the major presentation form of visualization (matrix or node-link diagram), the dimension of representation (2D or 3D), and the expression form of the time dimension (animation or timeline). Then the user interface is presented with the introduction of all the functions and views. At the end, the architecture of the visualization is illustrated, details of the demo system implementation will be introduced in this part.

Chapter 3 presents the two underlying algorithms with details. For the dynamic matrix seriation algorithm, I briefly review related works against different optimizing functions at first, then the algorithms are described by pseudo-code with examples for intuitive explanation. On the other hand, for the incremental community detection algorithm, previous outstanding approaches in this domain are reviewed as well. Then, the bipartition ration of community is defined and the algorithms are illustrated in details at the end.

Chapter 4 begins by introducing the background knowledge of the exploring method. It is explained that how this method is inspired by the chance discovery process via KeyGraph, including a brief introduction of both chance discovery theory and KeyGraph. Then, the pattern of potential discoveries on the visualization is defined combining with the exploring method.

Chapter 5 includes experimental evaluations and case studies for both underlying algorithms and the visualization. The matrix seriation algorithm is firstly evaluated against the famous Fisher's Iris data set by comparing with previous approaches. Then the experiment is introduced for evaluating the performance of our incremental community detection algorithm against the LFR benchmark networks by comparison with related works. Finally, the visualization is empirically evaluated through a human-centered study and a case study.

Chapter 6 concludes this dissertation. Firstly, contributions of this work are outlined and related to the challenges of online dynamic network visualization. Then, potential impact and application of this work are discussed within detailed scenarios. At the end, I discuss the research directions of this work in the future.

Chapter 2

Visual Interface

2.1 Design Features

The design is primary to implementation for visualization technology. Before developing the underlying algorithms and the exploring methods, what the visual interface looks like should be considered at first. According to the challenges of online dynamic network visualization, two goals are designed to be achieved by this work:

- Visualization should be intuitive and friendly to users, especially for tasks with online and dynamic features, because the time for users to understand the information delivered from the visualization is expected to be very short. Also, the observation made by users should be reasonable with the visualization, this can significantly improve the efficiency of further analysis based on the observation [9].

- Visualization should be economical, which means presenting more information by less cost. This cost includes the computational cost of the visual interface and the recognizing cost of users. In the case of visualizing dynamic networks in real time, we expect the visualization can deal with a large data stream of a network in a considerable speed.

For achieving these two goals, three important aspects of dynamic network visualization are considered in this work. With decisions of choices on these three aspects, a general image of the visualization can be provided for guiding the follow-up developments.

2.1.1 Matrix vs. Node-Link Diagram

The first problem considered for designing is which one should be chosen as the major presentation form in this work, matrix or node-link diagram? Solving this problem is relatively prior to the following two because the main expression form is the soul of a visualization technology. It decides the way of information delivering to users. The second and third aspects introduced in the following sections serve the first one for a better presentation.

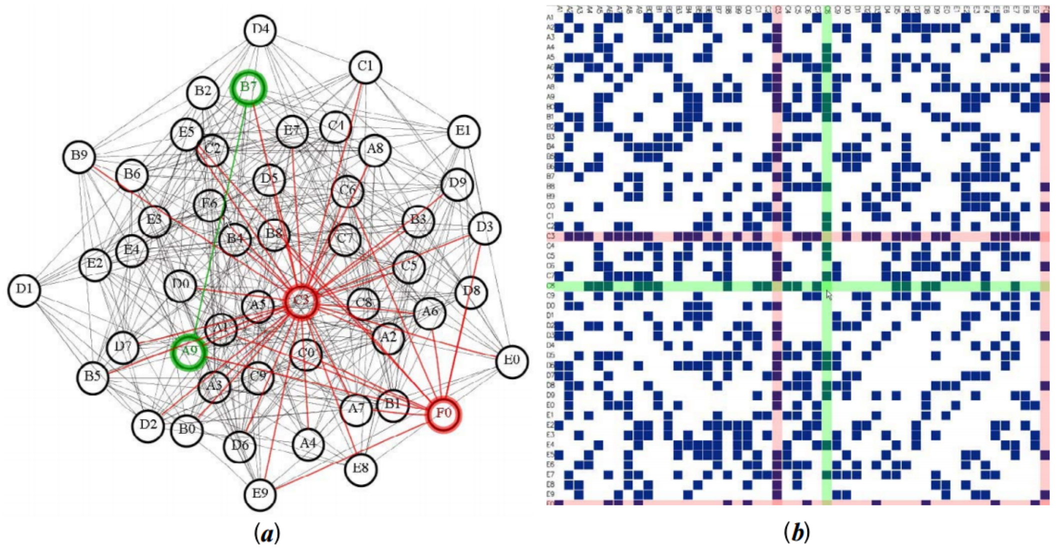

Figure 2.1: Visualizations of a network with 50 nodes and 400 links using different presentation forms: (a) node-link diagram, and (b) adjacency matrix

Figure 2.1 shows two presentations of the network published by Ghoniem et al. for comparing the readability of network representation between the node-link diagram and matrix [10]. It is difficult to conclude which one is better by simply looking at the figures. Actually, both of them show some advantages and disadvantages. For a node-link diagram, the most important advantage is the intuitive expression of the network. Node-link graph is the most natural way of representing a network since networks are originally graphs with additional features. Meanwhile, the drawback of the node-link graph is also obvious, that is the occlusion problem of graph layout. There are different types of occlusion on node-link diagrams: (1) occlusion on links by other links, which leads to mess and unreadable view when plenty of links cross together, (2) occlusion on links by nodes, which may result in misunderstanding of node belongingness and confusion of network structure as further consequence, (3) occlusion on nodes by other nodes, which causes difficulty of detecting individual nodes, and (4) occlusion on labels, which causes many problems related to labels, such as overlapping of labels, mis-belonging of labels (label of node A may be recognized as that of node B probability because of aggregated graph layout), and long label wrapping problem. Even it takes a high computational cost to avoid the occlusion problem, performance of the final appearance is still not guaranteed deal to specific dataset.

By contrast, there is nearly no occlusion problem on matrix layout. All the nodes are represented by matrix rows/columns respectively, and could be sorted based on orders such as the alphabetical order of labels, node degrees, semantical meaning, etc. Meanwhile, there is naturally no occlusion between links in the matrix, because all the links are represented by elements inside matrix without overlap. Also, labels of nodes are usually listed beside rows/columns for easy reference. However, drawbacks of the matrix also exist. Matrix layout is less flexible then graph layout, even when the latter is more complex. Elements (not order) within a matrix permutation should be exactly kept based on the network structure. As a result, when arranging a single element on position, the whole row/column containing this element will be relocated according to other rows/columns, while only two nodes need to be moved in node-link diagram for the same purpose. This difference leads to additional computation on arranging matrix layout, where matrix seriation algorithms aim at improving the efficiency of this computation.

Corresponding to the advantages and disadvantages of the matrix and node-link diagram, previous researches suggest that matrix presentation is more appropriate when the network is large and dense, while node-link diagram yields better performance on small and sparse networks [10]. Readability of both matrix and graph were evaluated on tasks such as quick node/link searching, degree counting, central nodes detecting and tracking, etc. Especially, experiments support the observation that analysis on complicated networks can benefit from the clear layout of the matrix, provided that matrix layout is well organized.

A Large number of previous approaches to dynamic network visualization employed node-link graphs as their major presentation form. Misue et al. firstly introduced a visualization method with dynamic adjustment approach of node-link diagram layout [11]. Brandes and Wagner introduced random field models for uniform representation by different layout models [12]. Instead of choosing from various layout models, Lee et al. regarded dynamic node-link graph layout problem as an optimization problem, which tried to meet different requirements by adjusting parameters [13]. On the other hand, some of the methods applied matrix instead. Bach et al. represented the matrix as a 3D cube, where the additional dimension represented the time-series [3]. Burch et al. employed a sequence of a bar chart on horizontal axis inside matrix cells to show the change of links [14], while Stein et al. applied the greyscale for the same purpose [15]. Also, Brandes and Nick introduced gestaltlines, a new expression of temporal changes on matrix elements [16].

2.1.2 2D vs. 3D

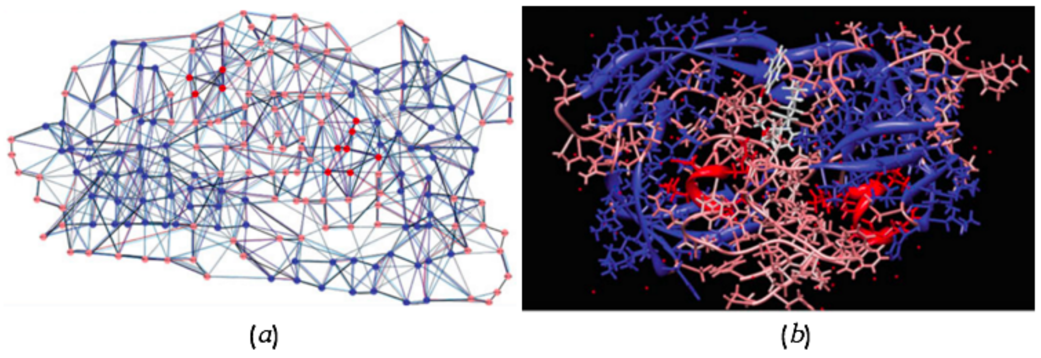

Figure 2.2: Visualizations of HIV-1 protease structure using different interface appearance: (a) 2-dimensional network, and (b) 3-dimensional netowrk

Figure 2.2 shows both 2-dimensional and 3-dimensional representation of

HIV-1 protease structure published by [17]. By separately considering expressiveness, 3D representation brings a view of the hierarchical structure from the outside in, comparing with the 2D representation where all the molecules lied on a single plane. 3D representations are more expressive by providing different perspectives of observation. Combining with zooming and navigating functions, data can be explored in very details. However, 3D representation suffers from some unavoidable problems as well, for example, the depth perception issue and more complicated navigation. In the case of dynamic network visualization, the most serious problem is the expensive computation of 3D shaping, edition, and animation.

By contrast, generating 2D representation is much faster and cheaper than 3D one for the same task. One of the drawbacks of 2D techniques is the visual overlap problem. As the previous section introduced, It always requires the additional cost of visualization for avoiding overlap and occlusion. Another potential problem is the learning cost of users. As we mentioned, 3D representation could be more intuitive than 2D one in many cases, but it still depends on specific cases. Previous research shows some experimental results that 3D visualization can significantly improve the user experience at the beginning of usage, although the best performance on 3D representation matched the 2D one only under the right combination of task and user [18].

Both 2D and 3D appearance of the visual interface have been employed in previous works. For example, Greilich et al. introduced TimeArcTrees aligning captures of network evolutions as 2D graphs [19]. Burch et al. employed 2D parallel coordinates for aligning [20]. Dwyer and Eades introduced a visualization that used 2D cylinders to represents nodes over time passage [21]. On the other hand, Erten et al. introduced a different way of performing top-down layout by using multiple surfaces in a 3D space as different time steps [22]. Federico et al. developed a demo system that predefined three views of 3D networks with fixed camera perspective and layer positions [23], while Itoh et al. released these settings to users [24].

2.1.3 Animation vs. Timeline

The major difference between dynamic network visualization and the static situation is the additional time dimension, which leads to new tasks of network analysis, such as detecting and tracking the evolution of network structure overtime. As the key feature of dynamic networks, temporal changes should be described clearly. Generally, either animation or timeline is employed for representing the temporal feature.

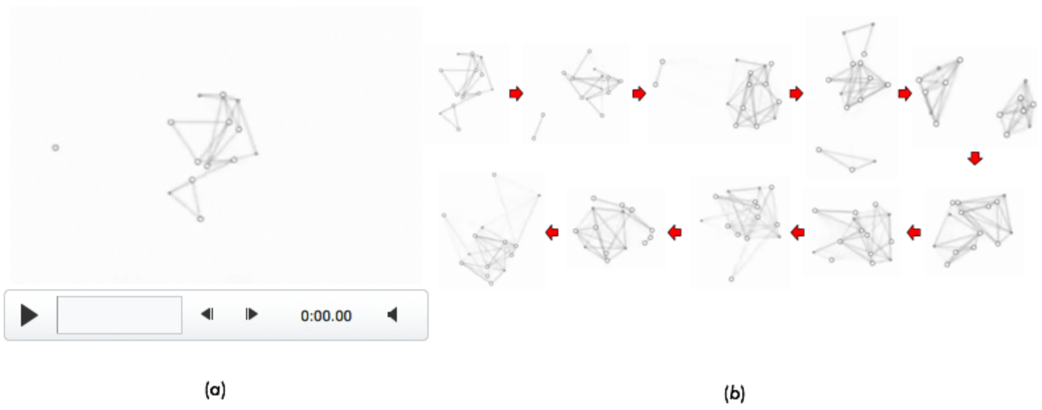

Figure 2.3: Visualizations of a network evolving process using different expressions of time series: (a) animation, and (b) timeline

Both animation-based and timeline-based visualization of the dynamic network are shown as an example in Figure 2.3. Different from the animation-based approach that only involves a single view of visualization, multiple views are listed as a sequence in the timeline-based visualization. Captures of presentation bring convenience on comparing network status between different time points, but with the drawback of difficulty in tracking tasks. Generally, it tends to be difficult to track the variation of a community or even a single node between captures, as the existence and the position of nodes and links may be unexpected because of the layout arrangement caused by network evolution. Also, a large number of captures generated from a long-term evolution may lead to the information overload issue. Also, detailed information has a high probability to be difficult for the user to observe when all the captures are packed together tightly in a single visual interface.

Different from timeline-based approaches, the whole evolving process of the network is presented in a single view in an animation-based approach. Relocation of nodes/links and evolution of network structure are described by animation so that temporal variation can be tracked easily following their actions. There was a qualitative study on the difference of user observation through animation-based and timeline-based representation. The experiment results indicated that representing temporal changes by animation encouraged users to make more findings on local changes and events, while more long-term discoveries were made through observing the timeline-based representation [25]. For the feature of instantaneity and rapidity, real-time systems always prefer short-term observations on the local area.

Many previous approaches employed animation-based representation for describing the time dimension. Diehl et al. introduced foresighted layout for stably presenting animation representation [26]. After that, Diehl and his colleagues extended the foresighted layout to focus on the adjustment strategies in different cases [27]. Instead of generating the super graph, Erten et al. connected equivalent nodes were connected by virtual links in their demo system named GraphAEL [28]. Some other approaches employed timeline representation. Ohsawa et al. implemented a tool for visualizing transient causes in time-series data [29]. Sugimoto et al. detected word clusters with temporally updated corpus using a sequence of word networks [30]. Also, virus-like visualization of node changes was represented by using timeline expression [22, 23].

2.2 User Interface

Based on the review of the three main aspects of dynamic network visualization in the previous section, we design the work to be a matrix-like, 2-dimensional, and animation-based approach in this dissertation. All the decisions are made for reducing the recognizing cost and the interface cost. By applying matrix as the major presentation form, large and complex networks can be described clearly without visual occlusion and mess view. Also, 2D appearance brings enough functions in most of the cases, and its efficiency on computation makes it more appropriate to online tasks. Furthermore, animation-based representation has been proved to perform better on local detecting and tracking tasks, which is also appropriate to the real-time visualization task.

Especially, adjacency matrix of communities instead of nodes is employed in this work. While visualizing large and dense networks, it is generally not necessary to present detailed topology for each node. Comparing with effect by a single node in a huge network, groups of nodes and their evolving trends attracts more attention from network analysts. That is community structure of the network, a key factor describing the macro-level topological structure of the network. Many types of research have attacked the problem of tracking community structure evolution on dynamic networks. For example, Vehlow et al. introduced a visualization improved from Sankey Diagram for reasoning network evolution [31]. Chihua et al. focused on brain networks and introduced a hybrid approach of node-link diagram and adjacency matrix [32]. Raghvendra et al. introduced Netgram that is also a visualization similar to Sankey Diagram with details like community size and modularity metrics [33].

The visual interface of the visualization introduced in this dissertation designed as Figure 2.4 shows based on the previous discussion. Simply, the interface involves two partitions, the control panel and the main view, where parameters, functions, and visual expressions are introduced in the following sections.

Control Panel

For simplifying the operation and learning cost by users, only essential functions are provided within the control panel, as the left part of Figure 2.4 shows. Especially, zooming function is originally provided by a web browser, so it is not involved in the system. Also, the restarting function of the visualization is purchased using the refreshing function of a web browser.

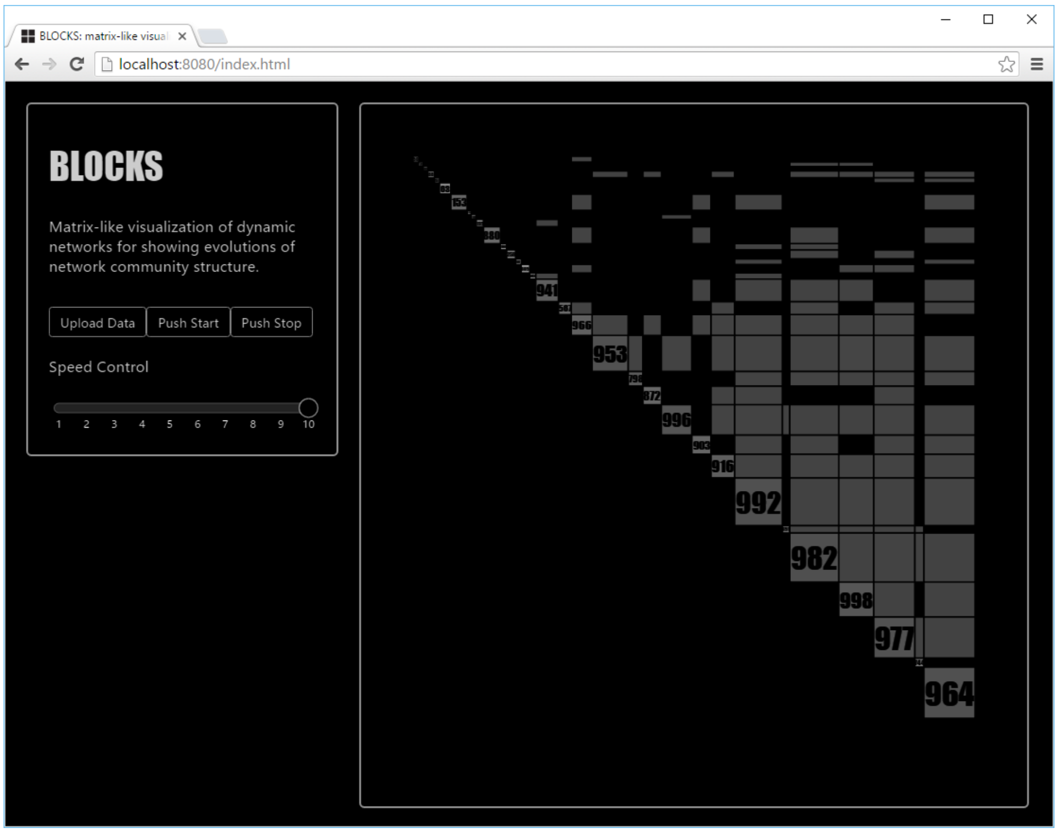
BLOCKS: matrix-like visual
localhost:8080/index.html
BLOCKS
Matrix-like visualization of dynamic networks for showing evolutions of network community structure.
Upload Data
Push Start
Push Stop
Speed Control
1 2 3 4 5 6 7 8 9 10
941
966
953
996
916
992
982
998
977
964

Figure 2.4: Visual interface design of the visualization introduced in this dissertation

Upload button responds to upload static database file to the system. This function is essential at this stage since the system is just a demo system, where the input data stream is simulated by sequentially receiving links from a static database. By clicking on this button, a file exploring window will appear and users can choose the dataset file for analysis. Input data file usually in the format of text, where each line represents a link between two end nodes. Links are ordered by their time stamps.

Start/stop button control the process of visualization. Start button can start/restart the process, while stop button can pause it. For the convenience of observation, pause is allowed in the demo system. However, it may not be possible in the real case, as data stream comes rapidly without pause.

Speed slider control the speed of the animation. The maximum speed is 20 links per second, while the minimum is 1 link per second. This function is also for the convenience of observation.

Main View

The matrix in this work is considered to be symmetric as we do not define the direction of inter-community relations. As a result, only half of the matrix is shown for reducing interface cost. There are two different kinds of elements. Those elements alongside the matrix diagonal represent communities. All the community elements are squares, where the element size represents the relative scale of the community comparing with all the other the communities, and the element opacity represents the graph density of community. Labels on those community elements correspond to central nodes within communities, where the centrality measure is node degree in this work. On the other hand, elements beside the diagonal represent the relations between communities. The opacity of the relation elements shows the connecting strength between related communities.

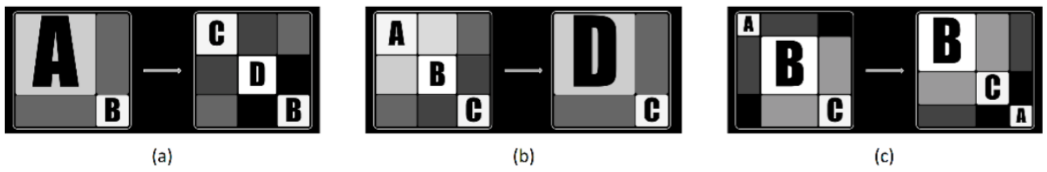

Figure 2.5: Representation of different events over the evolution of network community structure: (a) splitting, (b) merging, and (c) relocating

During the evolving process of the network, three categories of events are highlighted and described by the visualization for capturing temporal changes of network community structure. Figure 2.5-a shows a sample of community splitting process from community A to C,D. Especially, changes in relations are also described by our visualization according to changes on communities. Similarly, the merging process from community A and B to D is shown in Figure 2.5-b. The relocation of community A is shown in Figure 2.5-c. Strictly speaking, this relocating process is due to the dynamic matrix layout problem.

However, the rearrangement of matrix layout is caused by the evolution of network community structure, in this case, we regard relocation as the third category of evolving events on networks.

2.3 Architecture of the System

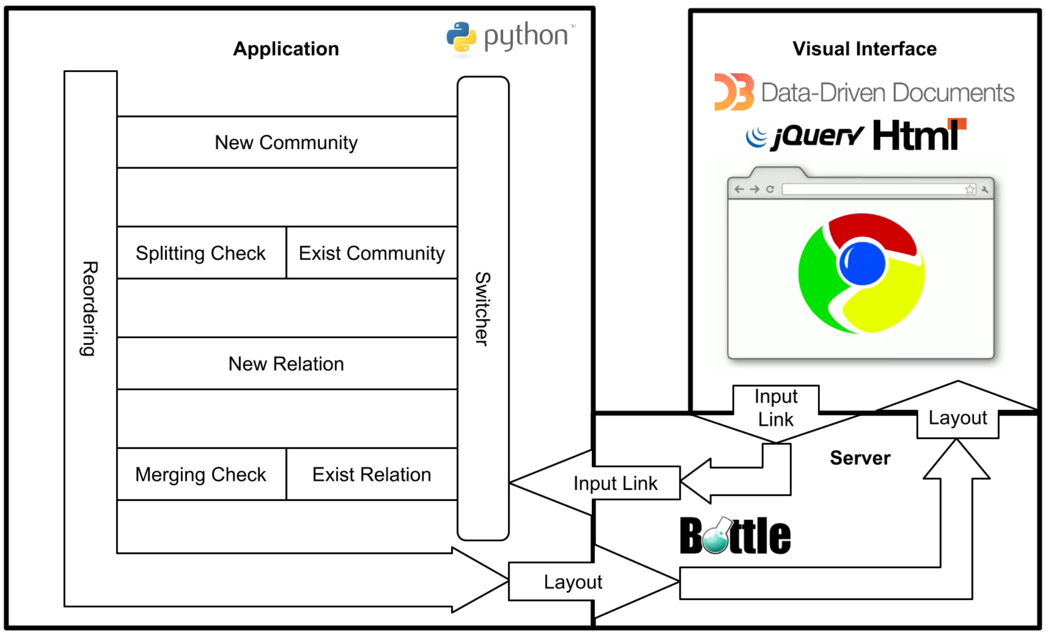

Figure 2.6: Architecture of the visualization system

For simulating the real online environment, the demo system is conducted on a web-service framework. There are three main partitions involved: the visual interface, the web server, and the background application. As mentioned in the previous section, the data stream is simulated by sequentially receiving data of single link from a static database. For every new coming link to the network, data is transmitted to the background application through the web server. After computing, the background application returns the revised layout information to the interface for an update. During this process, the web server transmits data by receiving, posting, and responding HTTP requests. Detailed descriptions are in the following sections.

Visual Interface

The main function of visual interface is representing the network evolution corresponding to the input link stream rapidly coming in real time. All the interactions between users and the system are held within this part. The new link data is packaged as a POST request and then sent to the server for further process.

The interface is implemented on a web browser using HTML/CSS for easy usage. All the animations and presentations driven by data are developed by D3.js, a widely used JavaScript Library for data-driven visualization. Also, JQuery is employed for data management and transmission from the front end.

Web Server

Web server is the transfer station of information in this system. It receives the new link data from the interface and transfer it to the background application, then post the layout data back to the interface. This part is essential for dealing with the rapid input in order.

We employed Bottle as the web framework for building the server. It is a simple and lightweight framework built on the standard Web Server Gateway interface. Bottle.py provides some useful functions like quick routing and file uploading, which helps simplify the developing works and improve the transfer efficiency.

Background Application

Both the matrix seriation algorithm and the community detection algorithm are implemented and executed in this part. After communicating with the web server, the new link is sent to a switcher with four branches:

- *New Community*: if both the two end nodes of the new link are new to the network, a new community is built based on this link.

- *Exist Community*: if one or both of the two end nodes belongs to an existing community, the new link will be added to the community, then it will be checked if this community should be split with the new coming link.
- *New Relation*: If the new link is between two communities originally with no relation, a new relation will be built based on this new link.
- *Exist Relation*: If the new link is built on two related communities, it will be added to that relation, and it will be checked if the two communities should be merged with the new coming link.

After adding the new link, the matrix layout updated and returned to the interface for further visualization. Algorithms are implemented using Python.

Chapter 3

Underlying Algorithms

3.1 Dynamic Community Detection Algorithm

For employing the adjacency matrix of the community as the presentation form, the evolution of network community structure should firstly be tracked, then an algorithm of dynamic community detection is essential. In this section, an incremental algorithm is introduced for dynamically detecting the temporal changes of both communities and relations between. In this section, previous methods are firstly reviewed by categories, then algorithm developed in this work is introduced in details.

3.1.1 Related Works

Based on different strategies adopted from different methods, most of the previous methods fall into two categories: optimizing-based approach and heuristic-based approach. The optimizing-based approaches find the optimal solution of the community detection problem by solving a equivalent optimization problem against a pre-defined objective function, for example, algorithms optimizing the network modularity defined by Newman [34, 35, 36, 37, 38, 39, 40, 41], the edge clustering coefficient [42], the maximization likelihood [43], the minimization of the Hamiltonian path [44], and the optimal compression of network [45, 46]

On the other hand, the heuristic-based approaches detect the network community structure by specifically designed rules or assumptions instead of transferring the community detection problem into an optimization problem [47, 48, 49]. Here, all the assumptions and rules are based on the recognition of network community, where the number of links inside communities should be larger than outside. Related works of both the two categories are reviewed in the following sections.

Optimizing-Based Approach

The most popular optimizing function is the Girvan-Newman modularity introduced by Girvan and Newman in their work in 2002, with the first algorithm of modern community detection on networks named the GN algorithm [34]. As a well known objective function, the GN modularity has been employed in many types of research. Clauset et al. introduced a fast greedy modularity optimization algorithm based on isolated nodes initially [35]. Also, some research performed a global optimization of modularity based on methods like simulated annealing [36, 37, 38, 39]. Blondel et al. introduced a local optimization algorithm of the GN modularity based on the neighborhood of each node and collected a considerable accuracy and linear computational complexity [40]. Another method by Donetti and Munoz introduced spectral properties of the graph (eigenvectors of the Laplacian Matrix) into the optimization of modularity [41]. This approach is more like a traditional clustering process in a multi-dimensional space.

Although GN modularity has been used as the quality function of community in many types of research, there is one issue that has been proved to limit the performance of algorithms optimizing GN modularity, that is the resolution limit [50]. This limit comes from the null assumption of modularity definition: each node is attachable on the network. However, this assumption has been proved to be unreasonable, especially for large networks. For avoiding this limit, some other functions were employed as the optimal function as well. For instance, Algorithm by Radicchi et al. is a divisive hierarchical algorithm similar to the GN algorithm, but optimizing the edge clustering coefficient instead of the GN modularity [42]. This coefficient was defined as the ratio between the number of circles involving an edge and the possible number of circles involving this edge. Also, two different types of communities were introduced with different coefficient conditions. Another work called expectation-maximization algorithm was introduced by Newman and Leicht, where the community structure is detected by Bayesian inference, and the best fit was inferred by maximizing a likelihood [43]. Also, for avoiding the resolution limit of modularity optimization, a model based on minimizing the Hamiltonian path was introduced by Ronbovde and Nussinov [44]. The optimal compression of the network structure was also employed for detecting the communities, as the structure should be recovered after decoding [45, 46].

Heuristic-Based Approach

The computational complexity of optimal algorithms generally tends to be high because of the global searching process. Even for a local search, it can still take a long time when the dataset is large. For this reason, some previous researches aimed to solve this problem through other means.Heuristic-based approaches aim to find a good enough solution in the reasonable time period, instead of indicating the best fit by optimization.

Even though the GN algorithm is based on the GN modularity, the algorithm itself is a heuristic algorithm. The algorithm performed a hierarchical division on links based on their betweenness. In the most popular implementation, the iteration of link division is stopped when the modularity of the current partition reaches the maximum [47]. Cfinder introduced by Palla et al. was built on the assumption of K-clique rolling community [48], the aim of this approach is for detecting communities that may overlap. Another approach simulating a peculiar diffusion process on the graph is introduced by Dongen et al [49]. The algorithm iteratively measured a probability matrix of a random walk on the network and enhance the matrix with a parameter α, until the matrix indicated a forest, whose components correspond to the communities.

3.1.2 Algorithm Design

We define the community detection problem in this dissertation as a dynamically splitting and merging problem. With the new links come, two changes may occur on the community distribution of the network, as Figure 3.1 shows. On one hand, one community may be split into two while there exist two groups of nodes inside this community that are linked denser inside and sparser between. On the other hand, if the relation between two communities become strong enough over time, then these two communities should be merged into one based on the change of network topology.

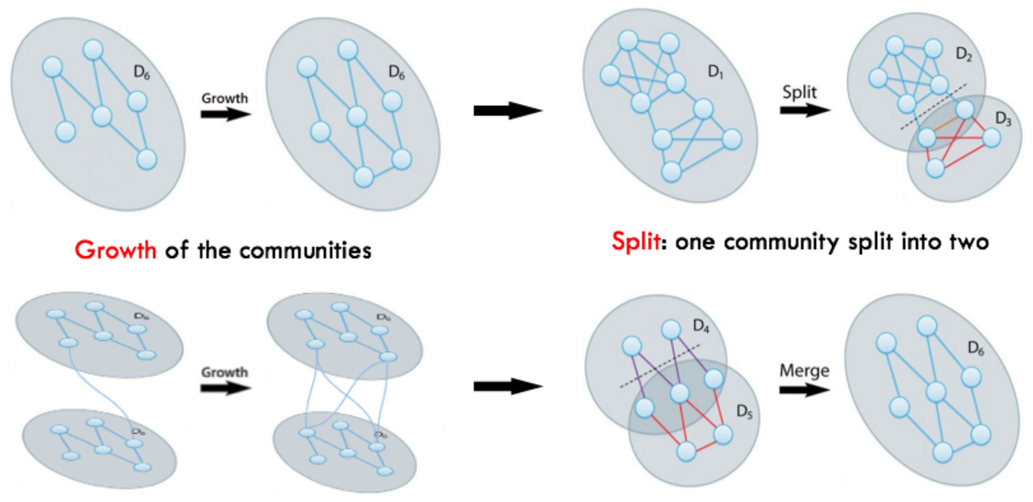

Figure 3.1: Splitting and merging conditions of communities within the evolution of network topology caused by the new coming links

Previous approaches focus on satisfying the cut criteria of network bipartition, which is always related to the number of links. The definition of modularity expects the number of links within communities to exceed the number expected by chance. As a result, intra-community links are dense and inter-community links are sparse. Heuristic-based approaches also follow similar rules and assumptions. A comparative study of community detection algorithms shows that both optimizing-based and heuristic-based algorithms underperformed when community structure was complex enough. Here, the complexity of community structure is evaluated by mixing parameter, which is defined as follows:

$$\mu = \frac{1}{|C|}\sum_{i\in C}\frac{k_i^{out}}{k_i^{out}+k_i^{in}} \tag{3.1}$$

Where i is a node inside community C, k_i^{out} is the number of i's neighbor nodes outside C, while k_i^{in} is the number of neighbors outside. In this case, simply considering the number of links is not enough because of the ignorance of group scale for both inside and outside the community. In other words, a large number of intra-group links do not respond to a dense group. For attacking this problem, we define the bipartition ratio for controlling the splitting/merging of the community. Given graph $G(V,E)$ and a bipartition X, Y of G satisfying the following conditions:

$$X\bigcap Y = \emptyset \quad and \quad X\bigcup Y = V \tag{3.2}$$

The bipartition ratio between X and Y is defined as follows:

$$\beta = \frac{N_{XY}}{|X|\times|Y|} \tag{3.3}$$

Where N_{XY} represents the number of links between X and Y. A smaller β leads to a higher graph density inside X and Y naturally because the total number of links inside G is fixed at the one-time point. The detecting problem is then transferred to minimization of β. However, even we have found the bipartition with the minimum β, the graph may be G still not appropriate to be split if the bipartition ratio is not small enough. We need a threshold, say θ_β, to clarify the splitting condition. With bipartition ratio and the threshold, we define the splitting/merging condition of community to be as follows:

- At time point t, a community $C(V_C,E_C)$ will be split into two communities $X(V_X,E_X)$ and $Y(V_Y,E_Y)$ if the following condition is satisfied:

$$\beta_C^t = \frac{N_{V_X V_Y}}{|V_X|\times|V_Y|} < \theta_\beta \tag{3.4}$$

- At time point t, two communities $X(V_X,E_X)$ and $Y(V_Y,E_Y)$ will be merged into one community $C(V_C,E_C)$ if the following condition is satisfied:

$$\beta_C^t = \frac{N_{V_X V_Y}}{|V_X| \times |V_Y|} \geq \theta_\beta \tag{3.5}$$

With the definition of bipartition ratio, an algorithm was developed for incrementally updating the bipartition ratio of communities every time a new link comes. For a community C, a bipartition X and Y of the node set of C, one end node of the new link v, and the current time t, the algorithm adds v to either X or Y and measures the bipartition ration for both the two situations as β_X and β_Y, then v is added to the side with lower value of bipartition ratio. Algorithm 1 shows the details.

Algorithm 1 Algorithm of Updating Bipartition Ratio

```
procedure BIPARTITE($C, X, Y, v, t$):
    $\beta_X \leftarrow C.Beta(X \bigcup \{v\}, Y)$  ▷ $C.Beta(X, Y)$: measure $\beta$ between $X$ and $Y$ on $C$
    $\beta_Y \leftarrow C.Beta(X, Y \bigcup \{v\})$
    if $\beta_X \leq \beta_Y$ then
        $\beta_C^t \leftarrow \beta_X$
        $X.add(v)$                              ▷ $X.add(v)$: add $v$ to $X$
        if $v \in Y$ then
            $Y.remove(v)$                       ▷ $Y.remove(v)$: remove $v$ to $Y$
        end if
    else
        $\beta_C^t \leftarrow \beta_Y$
        $Y.add(v)$
        if $v \in X$ then
            $X.remove(v)$
        end if
    end if
end procedure
```

The computational complexity of our algorithm is near $O(c)$, where c is a constant value. The main part of computation in the algorithm is $C.Beta(X, Y)$, with N_{XY} measured as:

$$N_{XY} = |E_C| - |E_X| - |E_Y| \tag{3.6}$$

Where $|E_C|$ represents the number of links in community C, and $|E_X|$ represents the number of links between nodes within X. By incrementally updating $|E_C|$, $|E_X|$ and $|E_Y|$, the algorithm can be very efficient as a whole. Details about running time of the algorithms are introduced in Chapter 5 based on the tracking records during the experiment.

3.2 Dynamic Matrix Seriation Algorithm

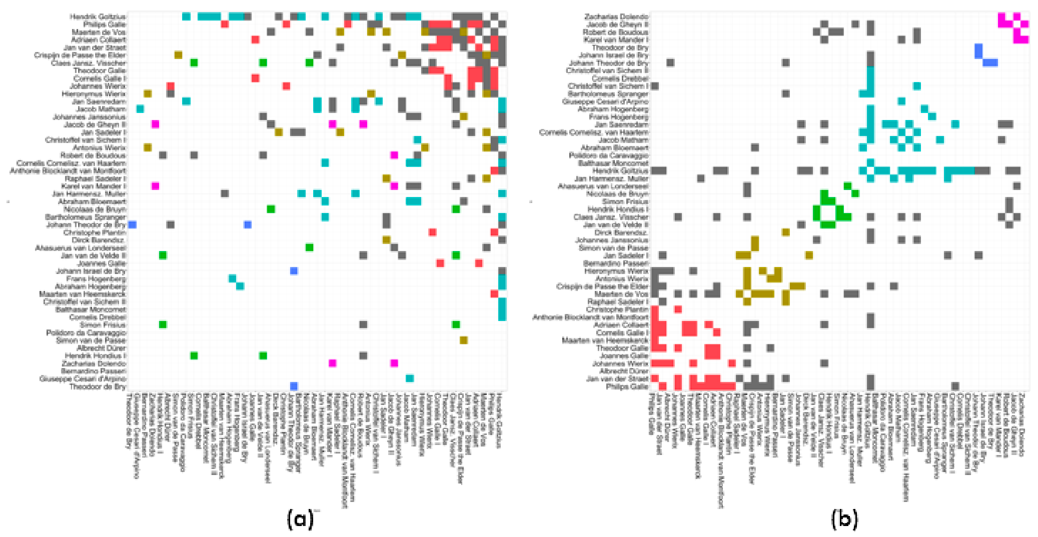

Figure 3.2: Matrix representation of the network of relationship between artists in Paris: (a) before seriation, and (b) after seriation

The dynamic community detection algorithm deals with the splitting/merging events of communities for describing network community structure over time. Also, the matrix layout should be dynamically rearranged for highlighting meaningful information hidden behind data. The seriation algorithm is developed for attacking this problem. Figure 3.2 shows the matrix presentation of a relationship network between artists in Paris before and after seriation [1]. It is difficult to detect useful information from Figure 3.2-a with a mess layout, while relations between artist groups are clearly described in Figure 3.2-b. Generally, a seriation algorithm seeks a matrix permutation by optimizing an

[1]Figures are originally published in http://matthewlincoln.net/2014/12/20/adjacency-matrix-plots-with-r-and-ggplot2.html

objective function. In this section, we firstly review the related works based on different optimizing objects, then we introduced the seriation algorithm we developed.

3.2.1 Related Works

Many optimizing functions have been defined in this domain, and they could be either loss function (the smaller the value is, the better), or metric function that is opposite to the loss functions. There are three widely used objective functions in this domain: the gradient measures, the least square criterion, and the Hamiltonian path length. Most of the previous works employed one of them or the alternative forms. We will introduce the previous works organized by these three measures, also some methods that do not belong to any of the three categories are introduced separately.

Gradient Measures

The gradient measures were defined based on the definition of the anti-Robinson matrix, where the row/column dissimilarity matrix contains monotonically nondecreasing values. In other words, for an anti-Robinson matrix with a set of row/column objects $\{O_1, O_2, ..., O_n\}$, this matrix holds the following gradient condition:

$$d_{ik} \leq d_{ij} \quad for \quad 1 \leq i < k < j \leq n \tag{3.7}$$

where d_{ij} represents the dissimilarity between O_i and O_j. For optimizing this condition, Hubert et al. [51] introduced the gradient measure as follow:

$$GM(O) = \sum_{i<k<j} f(d_{ik}, d_{ij}) \tag{3.8}$$

Where $f(x, y)$ is the sign function. Two of the sign functions are defined in [51]. The first one is the most traditional one like this:

$$f(x, y) = sign(x - y) = \begin{cases} +1 & if \quad x > y \\ 0 & if \quad x = y \\ -1 & if \quad x < y \end{cases} \tag{3.9}$$

And the second function is the weighted version of the first one, which is defined like this:

$$f(x, y) = |x - y| sign(x - y) = x - y \tag{3.10}$$

This formula gives an intuitive understanding of the gradient condition, that only cases following the monotonically nondecreasing order can obtain positive values. As a result, the gradient measure is a metric function (the larger, the better). Another similar criterion called anti-Robinson events used the same frame of the formula, but a different $f(x, y)$ function as follows [52]:

$$f(x, y) = I(x - y) = \begin{cases} 1 & if \quad x > y \\ 0 & otherwise \end{cases} \tag{3.11}$$

Some methods used the gradient measures as the optimizing criterion. Brusco and Stahl applied the branch-and-bound algorithm to the seriation problem, which can guarantee globally-optimal solution [53], but the scaling capability of the algorithm was limited up to size 35×35 because of the computational complexity. For improving the efficiency and the capability of matrix scale, Brusco et al. introduced a heuristic method based on dynamic programming escaping from local optima[54]. The simulated annealing was employed as the global optima for achieving this goal.

Least Squares Criterion

Another criterion for seriation is least squares criterion, which minimizes the divergence between the dissimilarity of two objects and their position gap.

This divergence is expected small, therefore the least squares criterion is a loss function (the smaller the better). Caraux and Pinloche introduced one formula to capture this divergence [55] as follows:

$$LS(O) = \sum_{i=1}^{n} \sum_{j=1}^{n} (d_{ij} - |i - j|)^2 \tag{3.12}$$

This formula uses the square error as the divergence measures, while another criterion called linear seriation criterion simply used multiplication instead [56], formula listed as follows:

$$LS(O) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}(-|i - j|) \tag{3.13}$$

Also, the 2-Sum loss criterion used similarity instead of dissimilarity in the formula [57], as the following formula shows:

$$LS(O) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{1 + d_{ij}} (i - j)^2 \tag{3.14}$$

Where similarity was simply defined as $1/(1 + d_{ij})$.

There are some methods using least square criterion or related metrics as the optimizing criterion. Rodgers and Thompson employed Multi-Dimensional Scaling to the seriation problem [58]. After obtaining an empirical order of the matrix by seriation, MDS was then used for scaling the two separate triangles of the proximity matrix defined by this ordering. Also, Ding and He introduced another approach of seriation by using spectral ordering [59]. The author showed a study on a k-way cluster assignment approach for releasing the limitation of special clustering methods applying on 2-way or k-way clustering problems.

Hamiltonian Path Length

By regarding the dissimilarity matrix of the object set as the adjacency matrix of a weighted graph, the seriation problem can be transferred to a Traveling

Salesman Problem (TSP). As a result, the Hamiltonian path length was introduced as an optimizing criterion of seriation problem [56]. The formula is as follows:

$$HP(O) = \sum_{i=1}^{n-1} d_{i,i+1} \tag{3.15}$$

Hamiltonian path length is also a loss function. There is a very good feature for this criterion, that is the only dissimilarity in the object order are needed for the measures. Gruvaeus and Wainer improved the hierarchical clustering algorithm and optimized the Hamiltonian path length [60]. For improving the effectiveness of algorithm, the author made the solution of clustering unique by introducing similarity between different solutions. Also, Sharlee and Wang introduced rearranging clustering, which was also based on TSP solution [61].

Other Approaches

There is also some other criterion for the seriation problem. For example, McCormic et al. defined the measure of effectiveness as the summation of the value product between one cell and all 4-way neighbors (up, down, left, and right) for all the cells in the matrix [62]. For an $n \times m$matrix $\mathbf{X} = (x_{ij})$, the measure of effectiveness is defined as follows:

$$ME(\mathbf{X}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}(x_{i,j+1} + x_{i,j-1} + x_{i+1,j} + x_{i-1,j}) \tag{3.16}$$

Then based on this thinking, Niermann defined another criterion named stress as the sum square error of the value between one cell and its (at most) 8-way neighbors [63], The local stress for element x_{ij} is defined as follows:

$$\mu_{ij} = \sum_{k=max(1,i-1)}^{min(n,i+1)} \sum_{l=max(1,j-1)}^{min(m,j+1)} (x_{ij} - x_{kl})^2 \tag{3.17}$$

and the global stress is defined as follows:

$$ST(\mathbf{X}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_{ij} \tag{3.18}$$

The famous algorithm optimizing measure of effectiveness is the bond energy algorithm (BEA) that is introduced by McCormic in the same paper of introducing the measure of effectiveness, while after two years, Lenstra proved that the optimization of BEA can be regarded as two independent TSPs [64].

3.2.2 Algorithm Design

There is a balance problem that most traditional algorithms are confronted with. Some approaches provide permutations with the perfect smoothness of the matrix permutation, where rows/columns are closely associated with the gradient order. However, smooth permutation may lead to the blurred representation of network structure, such as community structures and detailed structures inside communities. On the other hand, some other approaches highlight hidden information well by closer aggregating similar matrix rows/columns, but this aggregation is alway at the cost of losing necessary coherence of the matrix rows/columns.

Another problem of previous works is the limitation on the dynamic situation with the large matrix. Generally, seriation algorithms are designed for the small and static matrix, where no modification is involved over the time passage. However, in the task of dynamic network visualization, the matrix is generally large, and the adjacency matrix must be dynamic. As a result, an incremental seriation algorithm is essential in this case. Only one algorithm deals with this problem at the state of the art. Wittek introduced an incremental approach optimizing the Hamiltonian path length [65]. Seriation on matrix rows and columns were treated as two independent TSP problems. However, this approach only solved the problem of new coming rows/columns, because the core of the dynamic seriation in this approach was a quick searching and insertion based on a greedy strategy. For the situation of modification on existing rows/columns, the author did not provide a solution in the paper.

For attacking the problems stated above, we developed an incremental approach of seriation that maintains a globally optimal solution temporally by solving an optimization problem against the Hamiltonian path length. The algorithms involve two partitions, one is a local searching algorithm based on dynamic programming, another is a global optimizing algorithm based on simulated annealing. Following sections introduce the two algorithms in details with our designing philosophy.

Local Searching Algorithm

The local search algorithm is called "Recut" as for abstracting the process of recursively cutting and combining in the algorithm. This algorithm is based on a greedy strategy, that is greedily replacing high-dissimilar object pairs with low-dissimilar ones. We use dynamic programming to solve this problem as dynamic programming separates complex problems into simple sub-problems, and finally, combine the results of sub-problems to solve the complex root problem. The overlap problem is defined as follows: the algorithm cuts the object sequence into two sub-sequences at the object pair with the highest dissimilarity, then reorders the two sub-sequences separately and combines them back on the dissimilarity condition. In this case, the algorithm only needs to store the dissimilarity values for those object pairs that do appear in the permutation process. Instead of measuring the whole dissimilarity matrix at the beginning, our approach can definitely save the computing storage and time.

Algorithm 2 shows detailed information of Recut algorithm. For incrementally processing dynamic matrix, distances between objects are updated depending on demands dynamically. The updating rules are very simple: if the distance between two objects undertaken is not measured before, compute this distance and restore it in memory, otherwise, check if this distance should be updated based on the new coming link. The algorithm can be very efficient on theory. Time complexity of this algorithm is:

$$T = \begin{cases} O(c) & \text{if } n < 3 \\ O(n^L) + O(n^R) + O(c) & \text{otherwise} \end{cases} \tag{3.19}$$

Where n is the length of the input order sequence, n^L and n^R is the length

Algorithm 2 Local Search Algorithm

```
procedure RECUT($O$)                                ▷ $O$: n-length object sequence
    if $O.length < 3$ then
        return $O$
    end if
    $bp \leftarrow argmax_{i\in[1,n-1]}(d_{i,i+1})$         ▷ $d_{i,j}$: dissimilarity between $O_i$ and $O_j$
    $L \leftarrow Recut(O[1:bp])$             ▷ $O[i,j]$: sub sequence of $O$ from $O_i$ to $O_j$
    $R \leftarrow Recut(O[bp+1:n])$
    $cp \leftarrow argmin_{i\in\{l^L,r^L\},j\in\{l^R,r^R\}}(d_{ij})$ ▷ $cp$: combining point, $l^L, r^L$: the left and right end of $L$
    if $cp$ is $r^L, r^R$ then
        return $(L + Reverse(R))$   ▷ $L+R$: joint of $L$ and $R$ following the order shown
    else if $cp$ is $l^L, l^R$ then
        return $(Reverse(L) + R)$
    else if $cp$ is $l^L, r^R$ then
        return $(R + L)$
    else if $cp$ is $r^L, r^R$ then
        return $(L + R)$
    end if                          ▷ Totally 4 different conditions of combination
end procedure
```

of the two sub-sequences. $O(c)$ represents the constant time for running a fixed piece of code. The worst case of our algorithm is $n^L = 1$ or $n^R = 1$ for every iteration. In this case, the time complexity is $O(c(n-2))$, which is still linear. It was tested through experiments that the worst case appears quite rarely, and the algorithm can execute at a satisfying speed. Details about the experiments are introduced in Chapter 5.

Figure 3.3 shows an example of the running procedure of Recut. Here, we simply employ the Euclidean distance as the dissimilarity measure of the Hamilton path length. More detailed evolution of applying different distance formulas is presented in Chapter 5. Before running the algorithm, the matrix layout looks distributed, and it is hard to observe relations between matrix columns. Then After running Recut on columns of the matrix, the layout tends to become clearly and aggregately, where a continuous appearance of elements is presented intuitively, and it is reasonable this improvement comes from a significant decrease in the path length.

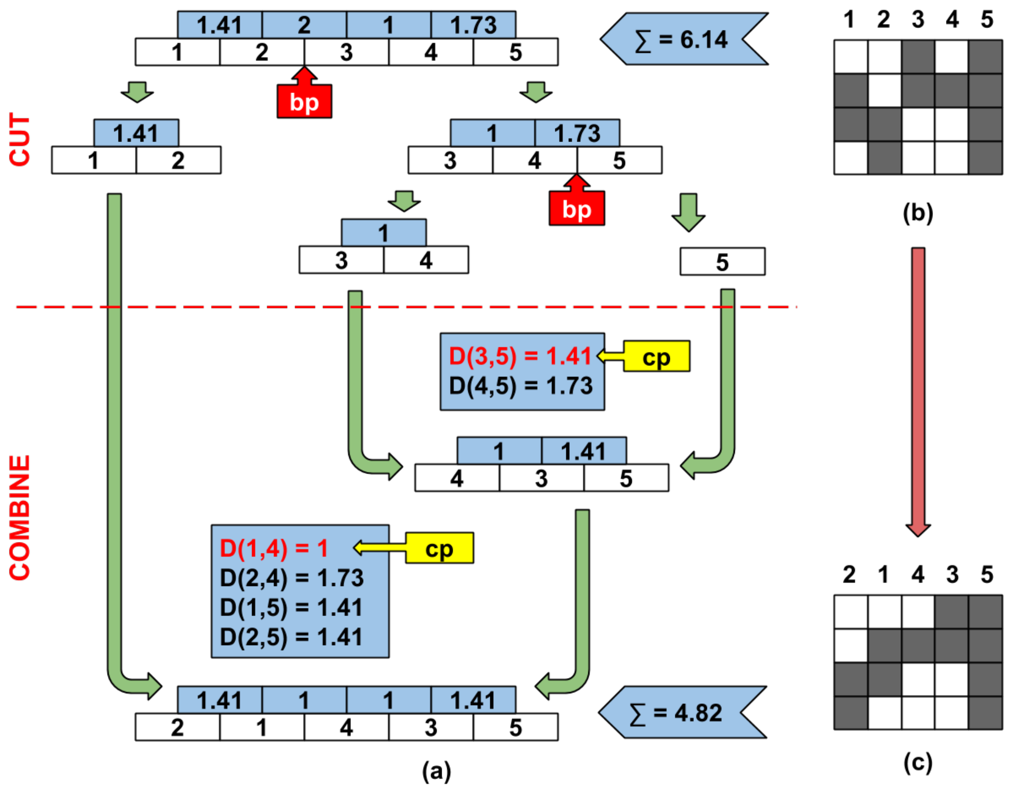

Figure 3.3: Illustration of the running procedure of the local search algorithm on sample matrix

As we mentioned in the previous section, considering both smoothing the permutation and highlighting important information together is a problem of seriation algorithms. The algorithm design of recursively cutting and combining is for attacking this problem. Previous approaches that optimized the Hamiltonian path length usually built the shortest path from the beginning with an initial object. By contrast, Recut arbitrarily set the current order as the initial state and searches for the shortest path by replacing long inter-distances with shorter ones. As a result, segments that are originally continuous are kept through the process of cutting, while segments that should be outlined together will be aggregated via combining.

Global Optimizing Algorithm

Generally, it is hard for algorithms based on a greedy strategy to achieve the global minimum, where the searching space is not a concave surface. As a result, a global optimizing algorithm is necessary. In our case, we choose simulated annealing as the technology for global optimization. Details of this algorithm are shown in Algorithm 3.

Algorithm 3 Optimization Algorithm

```
procedure OPTIMIZE($O$)                                  ▷ $O$: n-length object sequence
    for $t$ in $1 : t_{max}$ do
        $O^{new} \leftarrow recut(O)$
        if $Path\_length(O^{new} < Path\_length(O))$ then
            $O \leftarrow O^{new}$
        else                                             ▷ Case of achieving local minimum
            $O' \leftarrow randomly\_pick(N(O))$
            if $random(0,1) > P(t)$ then    ▷ $random(0,1)$: random number
in (0,1)
                $O \leftarrow O'$
            end if
        end if
    end for
    return $O$
end procedure
```

The end-loop condition of the algorithm can be set to either satisfying a threshold of the optimizing criteria or reaching a maximum cycle index. For seriation algorithms, there is no such a threshold of criteria as a standard. Therefore, we choose the maximum cycle index, notated t_{max}, as the end-loop condition. Every time when achieving a local minimum, the algorithm may choose to move to a neighbor state of current object order based on an acceptance probability. This probability is defined as follow:

$$P(t) = \frac{t}{t_{max}} \tag{3.20}$$

Where t is the current step number. This setting prefers to move to neighbor states in an early stage, but more inclined to keep the current state overtime. Here, the neighbor state space is defined as follows:

$$N(O) = \{O' | O' = \{O_1, ..., O_{i+1}, O_i, ..., O_n\},\ for\ i \in [1, n-1]\} \tag{3.21}$$

Where O is the current object order and O' is one of the neighbor states of O. Algorithm stops while achieving the end-loop condition t_{max}, and the best order, in the end, is returned. The temporal efficiency of this algorithm highly depends on the stopping condition t_{max}, which is pre-defined before running the algorithm. A parameter study of t_{max} is presented in Chapter 5, where we discuss the choose of t_{max} for either static and dynamic situations.

Noticed that the acceptance probability in simulated annealing algorithm is generally defined as follows:

$$P(T) = e^{\frac{-\delta E}{T}} \tag{3.22}$$

Where δE is the increase of energy (Hamiltonian path length in our case) from the previous time point to the current time, and T is another parameter named temperature for controlling the cooling-down procedure of the algorithm, calculated as follows:

$$T \propto \frac{t}{t_{max}} \tag{3.23}$$

However, we consider this setting to be inappropriate in our case. According to the design of the permutation algorithm, it is very easy to get $\delta E = 0$ (the new permutation is exactly the same as the old one while reaching a local optimization), which leads to an absolute acceptance of a random jump to a neighbor state, even when the cooling procedure is close to finish. For this reason, the δE is removed from the acceptance probability, and there is no need to employ the exponential function of $\frac{1}{T}$ anymore. Instead, the linear formula (Equation 3.20) is employed by experience for maintaining the slow decrease of acceptance probability over interaction.

Chapter 4

Exploring Method

4.1 Background

Generally, visualization technology is developed for accurately abstracting complex data, so that users can observe and understand the data in a more intuitive and readable way. Therefore, the most primary purpose of a visualization tool is an accurate representation of information. For this reason, previous research generally involves instructions about how the visualization describes target data. Especially for dynamic network visualization, most of the previous works provide illustrations about how to observe the network structures and the evolution over time with the visualization.

However, visualization technique itself is an important component of data mining and knowledge discovery. The final purpose of data visualization is for assisting further analysis in solving problems. Then a guidance on how to make meaningful observations on the visualization tool is also essential. With a clear instruction, users can quickly focus on the task of knowledge discovery without wasting time on considering what kind of observation has the potential to be meaningful. Motivated by this thinking, the exploring method for our visualization approach is illustrated for detecting observations that have the potential to be meaningful. This method is inspired by the procedure of chance discovery, where KeyGraph is employed as the key visualization technology for detecting chance from data. For better illustrating our method, we firstly give some background knowledge of chance discovery and KeyGraph.

Chance discovery was firstly introduced by Ohsawa as a process of discovering and reasoning a chance, where chance corresponds to events that are hard to be observed but have significant meaning for supporting decision making [66]. Based on the theory of chance discovery, chances always hide behind knowledge and observations that are obvious and common. For detecting chance from common knowledge, KeyGraph is employed as the key toolkit.

KeyGraph was firstly published as an automatic indexing approach of keywords in corpus independent from prior knowledge of semantics [67], and then introduced as an assisting technology of chance discovery [68]. A number of research on solving practical problems employed the scenario of chance discovery via KeyGraph. For example, Ohsawa introduced a work for detecting and reasoning risk activities during earthquake based on a sequence dataset of the earthquake in Japan [69], while working by Kenich et al. suggested the application of KeyGraph on product designing [70]. Seo et al. employed KeyGraph for chance discovery on online BBS, and experiments show a significant improvement of efficiency on team meeting [71]. Chance discovery via KeyGraph had also been employed to enhance genetic algorithms by detecting deeply hidden blocks [72].

Figure 4.1 shows what KeyGraph exactly looks like. There are two kinds of nodes with different colors in this graph. The black nodes represent common knowledge, which may relate to items frequently appeared in the dataset, while grey nodes represent less frequent ones. In KeyGraph, the black nodes conduct islands, which indicates communities with correlating common knowledge. And grey nodes are chosen as bridges between islands. From the most intuitive meaning, the grey nodes lead to events or knowledge which are rare but linking multiple general concepts together. On this meaning, information behind the grey nodes is detected as chance, which is reasonable based on the common knowledge linked by this chance.

The most major and recent application of chance discovery technique through KeyGraph is the Innovators' Marketplace of Data Jacket (IMDJ as

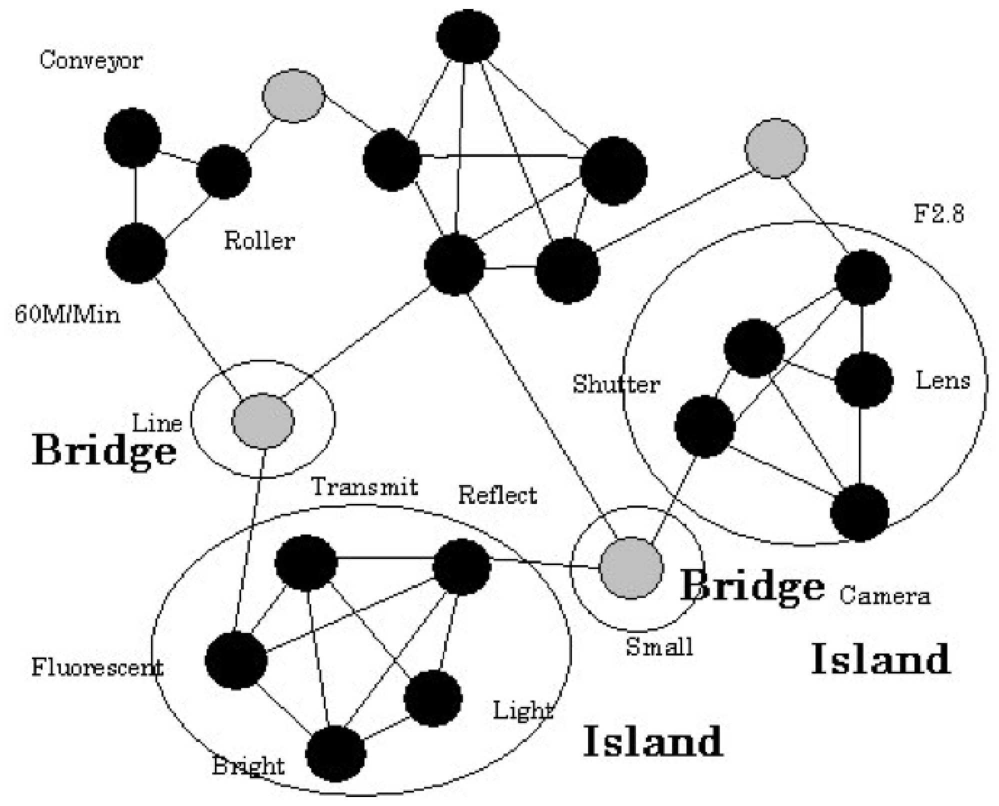

Figure 4.1: A sample capture of KeyGraph

short in the following paragraphs), which was systematically introduced in the work by Liu, Ohsawa, and Suda for the first time [73]. IMDJ is for providing a social environment to the data owners and potential users based on the market of data. There are two purposes of IMDJ, one is the validation of enclosed data by owners, and another is the innovation based on the market of data for solving practical problems. Here, what we said the market of data is different from the traditional data markets, such as Windows Azure Marketplace and KDnugget, which only provide a platform for trading data with a limited introduction of the contents of datasets without guarantee of the usefulness of the data to the buyers. On the other hand, the communication between expert and non-expert is generally difficult because of the big gap of knowledge background and experience, while IMDJ is also expected to provide the environment for enhancing this communication.

Figure 4.2 shows an sample sense of the IMDJ game published in [74]. There are two roles of participants in the game, the consumers provide the

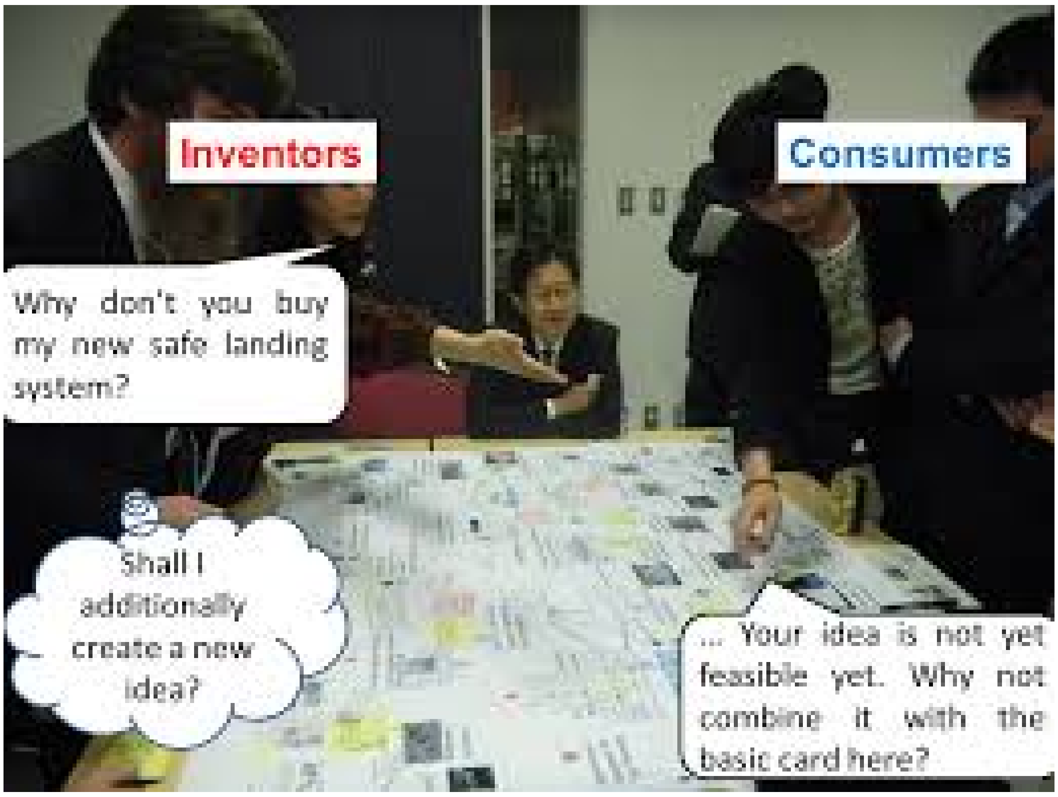
Inventors
Consumers
Why don't you buy my new safe landing system?
Shall I additionally create a new idea?
... Your idea is not yet feasible yet. Why not combine it with the basic card here?

Figure 4.2: The sense of the Innovators' Marketplace of Data Jacket as an example

requirements based on practical problems to be solved, and the inventors generate solutions to the requirements based on the KeyGraph and Dataset involved. Here, datasets are abstracted as data jackets, which includes the essential information for understanding the datasets without opening the data content. In this process, the chance discovery technique may be used for generating innovative solutions to the requirements. For example, the inventors may firstly search the bridge highly related to the requirement, and then combine the islands with data jackets to reason and finally generate the solution.

4.2 Procedure of observation

In our exploring method, some patterns of the visualization are defined for guiding users. Based on these patterns, users can easily detect events like birth and death of communities, splitting and merging of communities. Then, another pattern inspired by chance discovery via KeyGraph is defined for making potential meaningful observations.

Firstly, the birth and death of a community are remarked by a different color, as Figure 4.3 shows. In the case of the birth of a community, the element representing the new community will appear in green for a short period of time and then becomes white after that. On the other hand, red color remarks community that is going to dead, and then the community disappears from the matrix.

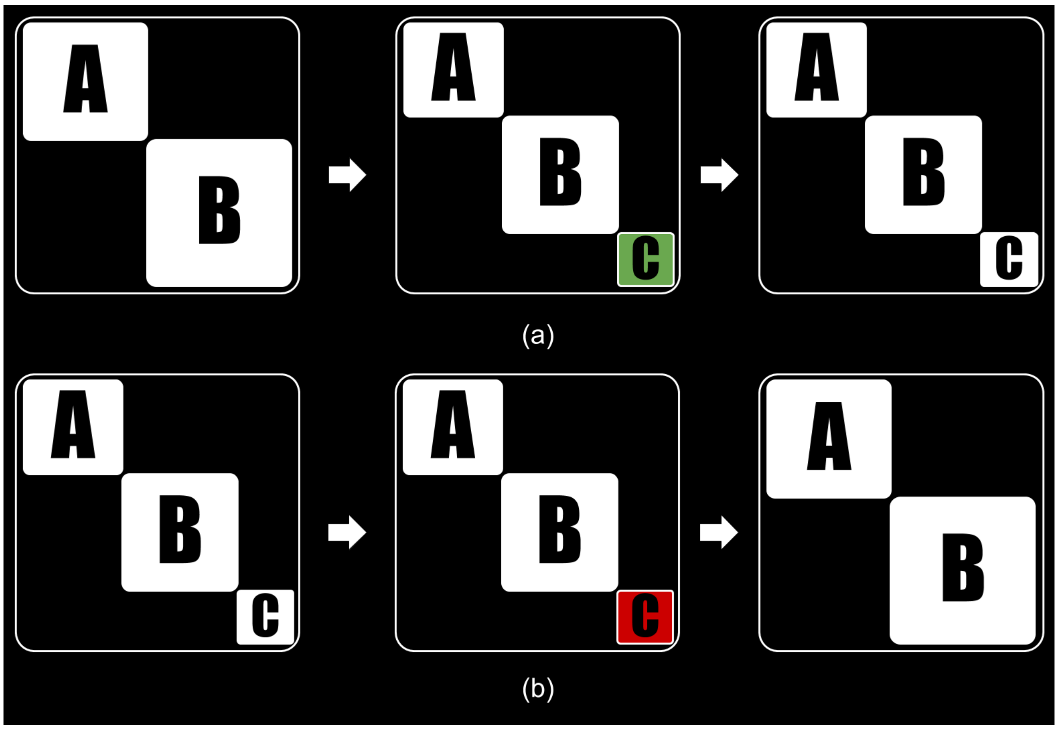

Figure 4.3: Remarks of birth and death of community: (a) birth, and (b) death

Another two events, community merging and splitting, are essential to be detected for describing the evolution of network community structure. Figure 4.4 shows the procedures respectively. At the beginning of merging, the two communities to be merged are marked by red color and then disappear. Instead, the targeting community appears in green and then recovers to white (maybe grey because of the low opacity). In contrast, the community going to be split is firstly marked by red and then disappears. At the same time,

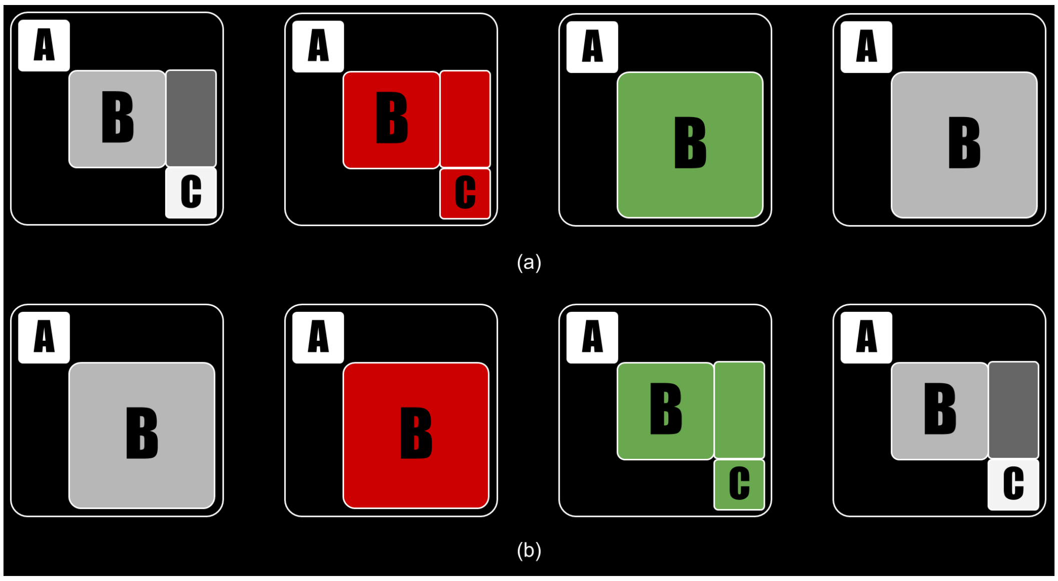

Figure 4.4: Remarks of splitting and merging of community: (a) merging, and (b) splitting

two new communities appear in green and then recover to white in a short period of time.

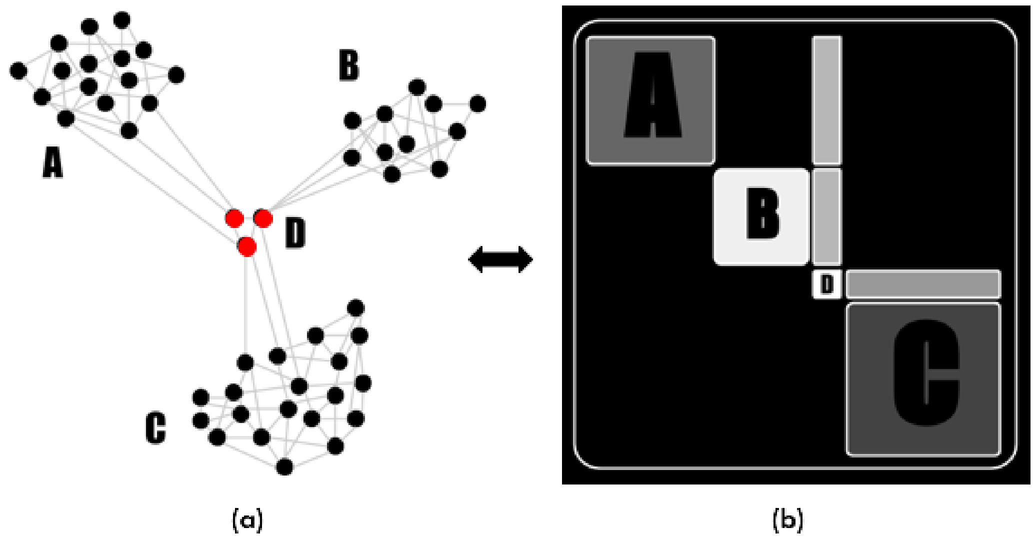

Figure 4.5: Pattern of potential meaningful discovery inspired by chance discovery via KeyGraph: (a) a sample network structure, where red nodes outline the rare but important event, and (b) representation by our visualization approach

One more pattern is defined for detecting potential discoveries. We extend the definition of the island in KeyGraph to a big and dense group of nodes in general networks, while bridge redefined by a small group of nodes linking at least two islands together. For example in Figure 4.5-a, this sample network shows an island-bridge layout, where group A, B and C are islands based on the extended definition, where all of them contains a number of nodes densely linked together. Differently, group D can be recognised as a bridge between A, B and C. In our visualization, a potential meaningful discovery is detected from a small community element connecting with at least two big community elements, as the community D shown in Figure 4.5-b. Especially, when matching this pattern in the real case, the matches may disappear after a period of time. We consider that better matches exist longer because longer appearance means a more reliable observation that is not observed by chance. However, there is no requirement on how long a match of the pattern should be. This threshold of the time length is decided by users in a specific case.

Chapter 5

Experimental Evaluation

5.1 Evaluation of Background Algorithms

Two parts of our visualization approach should be evaluated respectively for supporting our conclusion that our online visualization of dynamic networks is efficient. One is the evaluation of the underlying algorithms that completing all the automatic works for building the visualization. Another is the visualization presented by the user interface, which drives all the interactions from users during the exploring and discovering process. This section presents the performance evaluation of the two underlying algorithms introduced in Chapter 3. Several experiments against either real data set or sample data generated by benchmark algorithms are conducted with statistical analysis and discussion of the results.

For the community detection algorithm, a parameter study is presented for learning the setting of θ_β as the threshold of bipartition ratio as the splitting/merging condition. Then, I discuss the contributions of our algorithm through a series of comparisons with previous works. Comparisons are based on networks generated by a famous benchmark algorithm in this domain.

For the seriation algorithm, two parameter studies are firstly conducted for exploring the t_{max} setting in the global optimizing algorithm and the influence of choosing different distance formula in the local search algorithm. Then, another experiment is presented for evaluating the algorithm performance comparing with some previous outstanding algorithms of matrix seriation, with a discussion about how our algorithm brings significant improvement to previous works.

5.1.1 Dynamic Community Detection Algorithm

In this section, experiments are conducted for evaluating the performance of the dynamic community detection algorithms comparing with previous methods. Before that, the threshold of bipartition ratio θ_β is learned by a parameter study. Evaluating criteria employed in the experiments is the Normalized Mutual Information (NMI), which has been widely used for evaluating community detection algorithms. The reason of not using modularity as the criteria is because the well-known resolution limit of modularity, which often leads to inaccurate descriptions of larger networks and smaller communities [50].

In this dissertation, the definition of NMI employed is published by Strehl and Ghosh [75], where NMI is calculated akin to the Pearson correlation coefficient. The formula is defined as follows:

$$NMI(GT, CD) = \frac{I(GT, CD)}{\sqrt{H(GT)H(CD)}} \tag{5.1}$$

Where $GT = \{C_1, C_2, ..., C_{k^{GT}}\}$ represents the ground truth of community distribution in network G, and $CD = \{C'_1, C'_2, ..., C'_{k^{CD}}\}$ represents the community detection result. C and C' are node sets representing communities, while k^{GT} and k^{CD} are the number of communities in GT and CD. Here, k^{GT} does not have to be equal to k^{CD}. $I(GT, CD)$ represents the mutual information between GT and CD, which is defined as follows:

$$I(GT, CD) = \sum_{h=1}^{k^{GT}} \sum_{l=1}^{k^{CD}} n_{h,l} log(\frac{n \times n_{h,l}}{n_h^{GT} \times n_l^{CD}}) \tag{5.2}$$

Where n is the total number of nodes in network G, $n_{h,l}$ is the number of nodes belong to both the h^{th} community of GT and the l^{th} community of CD at the same time, n_h^{GT} represents the number of nodes inside the h^{th} community of GT, and n_l^{CD} is the similar. $I(GT, CD)$ describes how much information are shared by GT and CD, but this measure is not idea as the criteria. With formula 5.2, all the partitions of GT as CD will lead to the same result as $I(GT, CD) = H(GT)$, even though these partitions are totally different. Here, $H(GT)$ is the entropy of GT defined as follows:

$$H(GT) = \sum_{h=1}^{k^{GT}} n_h^{GT} log \frac{n_h^{GT}}{n} \tag{5.3}$$

To avoid that, $\sqrt{H(GT)H(CD)}$ is introduced for normalizing the mutual information, as formula 5.1 shows. NMI equals 1 if GT and CD are exactly the same, whereas the value is expected to be 0 if GT is totally independent of CD.

Parameter Study: learning the threshold of bipartition ratio θ_β

For learning an appropriate setting of θ_β, performance of the algorithm was tested by running on a sample network while varying the θ_β value. The network was generated by the famous Lancichinetti-Fortunato-Raddichi (LFR) benchmark algorithm, which has been widely used for evaluating community detection algorithms [76]. The sample network was generated by employing the following parameters of the benchmark algorithm: 20 for the average degree, 50 for the maximum degree, 2 for the minus exponent of degree distribution, 1 for the minus exponent of community distribution, and 0.1 for the mixing parameter μ. The total number of nodes inside the network was set to be either 1000 or 5000. Also, two ranges of community size in networks were set as follows: S notated $10 \sim 50$ nodes in a community, and B notated a larger size of $20 \sim 100$ nodes. As a result, there were four settings of the network: $1000S$, $1000B$, $5000S$, and $5000B$.

θ_β varied from 0.2 to 0.8 with an offset of 0.05, then there were totally 13 settings of θ_β. For each value of $\theta_\beta$and each setting of the network, our algorithm was executed for 100 times and average NMI was measured for each case. This result is shown in Figure 5.1 as a line chart. A peak value can be indicated at $\theta_\beta = 0.5$ in general, although this trend is not significant and the performance at $\theta_\beta = 0.55$ is even slightly better in case of $5000B$. This result matches the expectation. The Too small value of θ_β may lead to over-splitting, while too large value causes over-merging. Algorithm in both of the two cases will underperform and result in a relative worse performance on NMI. Based on this result, we conclude that $\theta_\beta = 0.5$ is an appropriate setting, and we employ this setting to the following experiments.

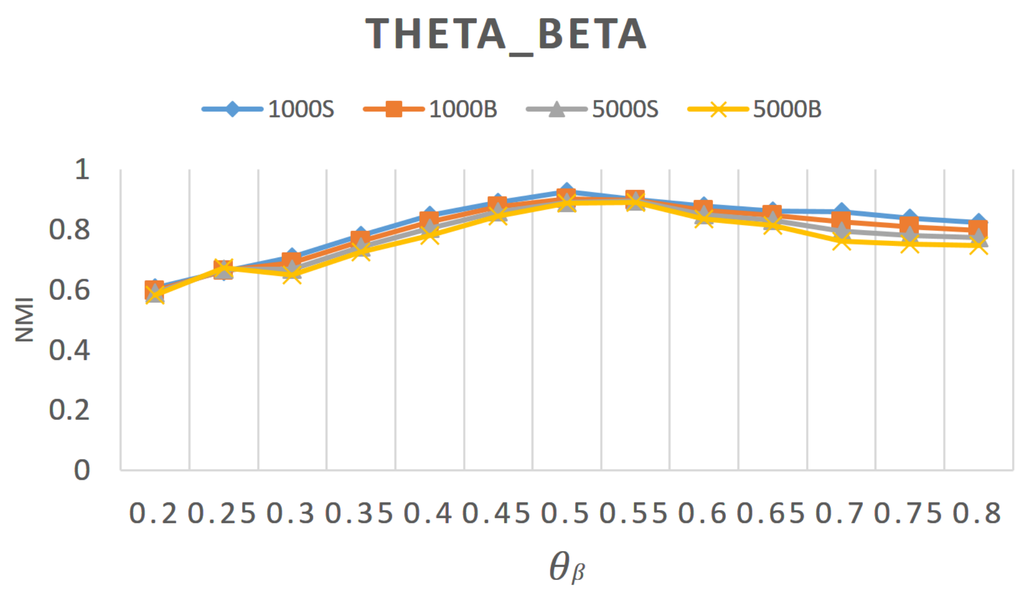

Figure 5.1: NMI result for learning θ_β, the threshold of bipartition ration

Performance Study: comparing with previous outstanding algorithms

For testing the improvement, an experiment was conducted for comparing algorithm in this work with a number of previous outstanding approaches. Networks for evaluation in this experiment was generated by employing nearly the same parameter setting of the LFR benchmark algorithm as the previous parameter study, but the mixing parameter varied from 0.1 to 0.8 with an offset of 0.05. Table 5.1 lists all the methods for comparison:

Evaluation results of these algorithms were published by Lancichinetti and Fortunato in their work of a comparative study between community detection algorithms [77]. Actually, there were some other algorithms involved in this

Table 5.1: Previous outstanding algorithms of community detection for comparison

Name	Type	Abbreviation
Greedy Optimization [35]	Optimizing-Based	CLAUSET ET AL.
Fast modularity optimization [40]	Optimizing-Based	BLONDEL ET AL.
Algorithm by Radicchi [42]	Optimizing-Based	RADICCHI
Potts model approachv[44]	Optimizing-Based	RN
Maps of random walks [46]	Heuristic-Based	Infomap

study, but these five algorithms were highlighted with their good performance by the authors. Then, the comparison was performed against these five approaches with results shown in Figure 5.2, where algorithm in this dissertation is notated by "BIPARTITE".

Four algorithms, Informap, RN, BLONDEL ET AL., and RADICCHI, generated the community distribution exactly the same as the ground truth with μ varied from 0.1 to 0.35. Especially, Infomap and RN led the performance by maintaining $NMI = 1$ even with the very complex network where $\mu = 0.55$ for all the cases. The author of work [77] also summarized that Infomap and RN topped the performance in this study, while the performance of BLONDEL ET AL. was also considerable with a high level of NMI till $\mu = 0.65$ for all the network settings. By contrast, CLAUSET ET AL. performed the worst among all the algorithms, where a sharp decrease of the curve was shown for every case of a network in Figure 5.2-e. Although RADICCHI yielded a good performance at the early stage, it could still be seen that the NMI value decreased abruptly when μ reached 0.5. After that, the NMI values stayed stable at 0 till $\mu = 0.8$ for all the network settings.

Comparing with the five previous approaches, there was an important improvement by BIPARTITE, that was the stability of performance in case of complex networks. It could be seen clearly that previous algorithms yielded unsatisfying performance when networks were complex with $\mu \geq 0.65$, even for Infomap and RN that provided outstanding performance when $\mu < 0.65$. In contrast, the performance of BIPARTITE is more stable, where NMI value varied from 0.8 to 0.9 before μ reached 0.45, and maintained greater than 0.5

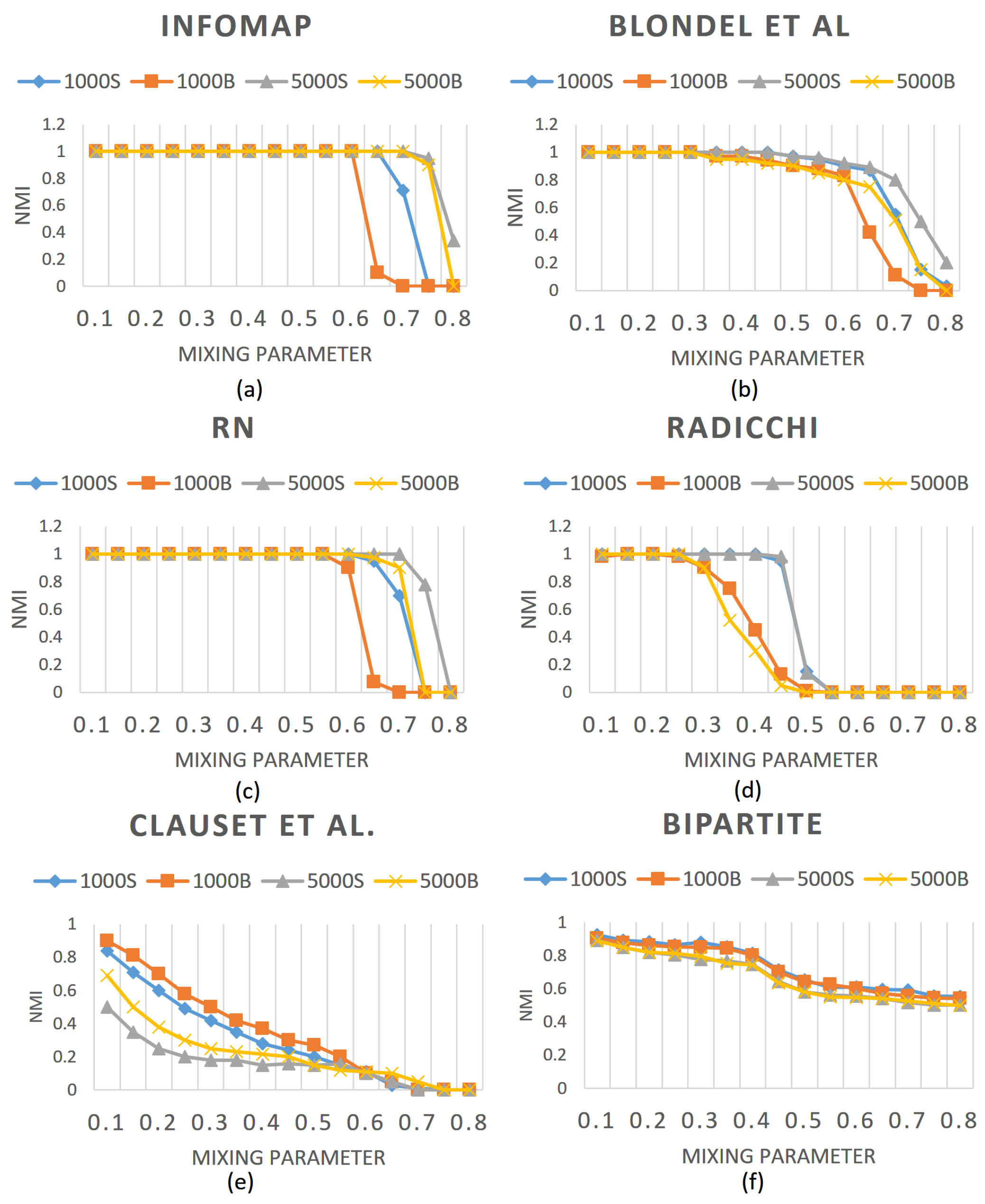

Figure 5.2: NMI result of both our algorithm and previous outstanding algorithms for comparison: (a) Informap, (b) BLONDEL ET AL., (c) RN, (d) RADICCHI, (e) CLAUSET ET AL., and (f) BIPARTITE

even for the most complex case of $\mu = 0.8$. Meanwhile, there was no obvious gap between the four cases of the network for BIPARTITE, while a big gap was shown by RADICCHI and CLAUSET ET AL.

For the running time of the algorithm, the computational complexity of both Infomap and BLONDEL ET AL are essentially linear in network size, which is the fastest among all the algorithms. The method RN provides a relative worse time complexity (superlinear in a number of links in the network) than Infomap and BLONDEL ET AL, but better performance than our algorithm with a linear complexity on a number of links. Then comes to CLAUSET ET AL., which can achieve a complexity of $O(Nlog^2N)$ with the efficient data structure, and RADICCHI has the complexity of $O(N^2)$. However, this ranking is for static community detection. In the dynamic case, our algorithm can top the performance with a nearly constant complexity of computation. We recorded the time usage of our algorithm and found that it only took 0.0015 seconds in average for an update of community structure with a new link coming.

5.1.2 Dynamic Matrix Seriation Algorthm

This section presents all the experiments for learning and evaluating our matrix seriation algorithm. Based on the brief review of seriation algorithms in Chapter 3, we choose the gradient measure, the least squares criterion, and the Hamiltonian path length as the evaluating criterion. Table 5.2 lists the formulas for convenient reference. All the comparisons and evaluations in this section are all based on these three formulas.

Table 5.2: Formulas of the evaluating criterion of seriation algorithms

Name	**Type**	**Formula**
Gradient Measure	Metric	$GM(O) = \sum_{i<k<j} f(d_{ik}, d_{ij})$ for $f(x,y) = sign(x-y) = \begin{cases} +1 & if \quad x > y \\ 0 & if \quad x = y \\ -1 & if \quad x < y \end{cases}$
Least Square Criterion	Loss	$LS(O) = \sum_{i=1}^{n} \sum_{j=1}^{n} (d_{ij} - \lvert i-j \rvert)^2$
Hamiltonian Path Length	Loss	$HP(O) = \sum_{i=1}^{n-1} d_{i,i+1}$

Parameter Study 1: learning t_{max} in the global optimizing algorithm

Firstly, experiment for learning t_{max} is conducted. The experiment setup is simple, executing the seriation algorithm with different values of t_{max}, it is expected to detect an appropriate value instead of an arbitrary setting for the further experiments. t_{max} was varied from 100 to 1000 with an offset of 100, so there were totally 10 settings of t_{max}. For every parameter setting, the algorithm was executed for 100 times, and the three criterion were measured for each time. Finally, the average values of the criterion were measured for each setting of t_{max}. Especially, the distance formula in the local search algorithm was fixed by employing the Euclidean distance for controlling variables, and another experiment was presented in the following section for learning the setting distance formula. Dataset employed in this experiment was the famous Fishers Iris dataset, which was firstly published by Fisher [78]. Matrix transferred from this dataset was a 150×150 symmetric matrix, where rows/columns represented cases of iris, and elements of the matrix corresponded to the similarity between different iris case pairs.

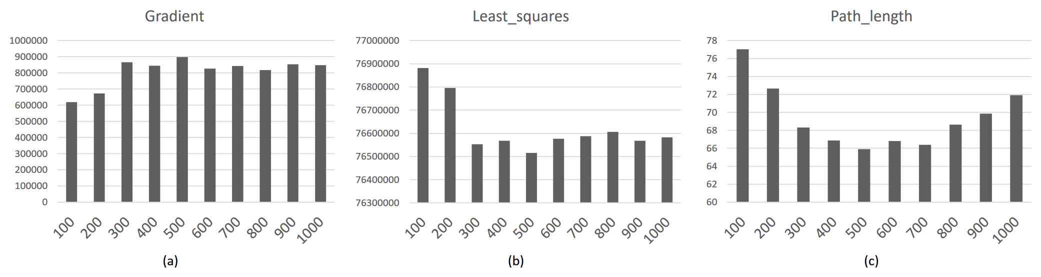

Figure 5.3: Parameter study result of t_{max} in global optimizing algorithm against three evaluating criterion: (a) Gradient Measure, (b) Least Square Criterion, and (c) Hamiltonian Path Length

Figure 5.3 shows a bar chart representing the learning result of the algorithm performance based on the variation of t_{max} value. In Figure 5.3-a, the higher the bar is, the better result it represents since the gradient measure is a metric function. It can be seen that $t_{max} = 500$ brings the best performance comparing with the others, where $t_{max} = 100$ is the worst. Although there seems to be no big difference of performance from $t_{max} = 300$ to $t_{max} = 1000$

in the figure, the gap between the real values of the gradient measure is large, which means setting $t_{max} = 500$ leads a significant better performance than other settings. On the other hand, $t_{max} = 500$ tops the performance of both the least square criterion and the Hamiltonian path length, with smaller values of loss function than any other setting.

In general, a peaking trend with the variation of t_{max} value can be seen, where $t_{max} = 500$ tops the performance, and then gets worse and worse to both directions. This trend appears significantly in 5.3-c. It can be explained by the theory of simulated annealing. if t_{max} is too small, the process of global searching will be ended too early, where it has a high probability that the global optimization is not reached yet. By contrast, a too big t_{max} leads to a higher probability to transfer to a random neighbor state even after reaching the global optimization.

It should be emphasised that this experiment is built on a static case, where the matrix does not change over time. In the case of a dynamic matrix, the process of global optimization is even shorter, where t_{max} generally tends to 50. This is because the change for each step is small and has little effect on the matrix layout, and the algorithm can reach the global optimization in just a few steps based on the optimizing result from the previous step.

Parameter Study 2: choosing distance formula in the local searching algorithm

For exploring the effect of distance metrics on the local search algorithm, we compared the algorithm performance by employing different distance formulas. Table 5.3 shows all the distance metrics considered in our experiment, where $\mathbf{p}$ and $\mathbf{q}$ represent two n-length vectors, and $d(\mathbf{p}, \mathbf{q})$ represents the distance between. Also, p_i represents the i^{th} element in vector $\mathbf{p}$, and μ_p represent the expectation of all the values in $\mathbf{p}$.

The experiment setup was similar to parameter study 1. For every distance metric, the algorithm was executed for 100 times, and values of all the three criterion were recorded for each time, and average values were computed. The

Table 5.3: Formulas of distance metrics for comparison

Name	**Formula**
Canberra	$d(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{n} \frac{\lvert p_i - q_i \rvert}{\lvert p_i \rvert + \lvert q_i \rvert}$
Bray-Curtis	$d(\mathbf{p},\mathbf{q}) = \frac{\sum_{i=1}^{n} \lvert p_i - q_i \rvert}{\sum_{i=1}^{n} \lvert p_i + q_i \rvert}$
Euclidean	$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$
Cosine	$d(\mathbf{p},\mathbf{q}) = 1 - \frac{\sum_{i=1}^{n} p_i \times q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \times \sqrt{\sum_{i=1}^{n} q_i^2}}$
Correlation	$d(\mathbf{p},\mathbf{q}) = 1 - \frac{\sum_{i=1}^{n} (p_i - \mu_{\mathbf{p}})(q_i - \mu_{\mathbf{q}})}{\sqrt{\sum_{i=1}^{n} (p_i - \mu_p)^2} \sqrt{\sum_{i=1}^{n} (q_i - \mu_q)^2}}$

Fisher Iris data set was also employed in this study, where the t_{max} was set to be 500.

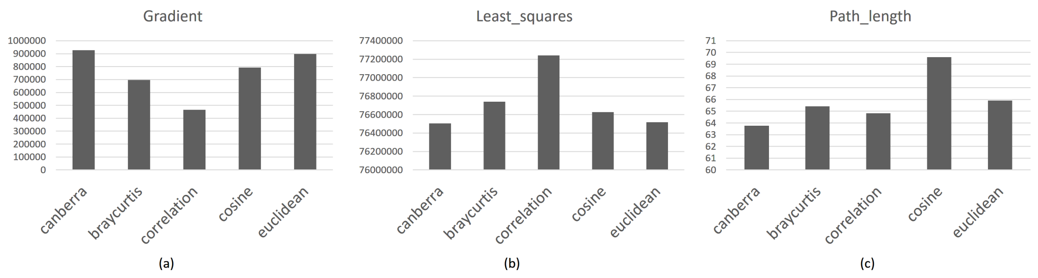

Figure 5.4: Parameter study result of distance metric in local searching algorithm against three evaluating criterion: (a) Gradient Measure, (b) Least Square Criterion, and (c) Hamiltonian Path Length

Figure 5.4 shows a bar chart as the comparing results, where values are the average value in each case. For the gradient measure, both Canberra distance and Euclidean distance obtain better performance than others, while Canberra distance performs slightly better than Euclidean distance. By contrast, the correlation dissimilarity performs the worst among all the metrics. The result is similar to the least square criterion as Figure 5.4-b shows. On the other hand, Canberra distance provides a significantly better performance than other metrics, and the worst is the cosine dissimilarity in this case. Based on this result, we conclude that Canberra distance leads to better algorithm performance for all the three optimizing objectives.

Performance Study: comparing with previous outstanding algorithms

For proving and exhibiting the improvements of the algorithm in this dissertation, a performance study against the Fisher Iris data set was conducted by comparing with 5 outstanding algorithms that have been proved to be useful. Table 5.4 lists the algorithms for comparison with details. Especially, all the methods are implemented by applying an R package named "Seriation" [79].

Table 5.4: Previous outstanding algorithms for comparison

Algorithm	Abbreviation	Optimizes
SImulated Annealing Heuristic [54]	ARSA	Gradient measure
TSP solver by Climer and Zhang [61]	C&Z	Hamilton path length
Multi-Dimensional Scaling [58]	MDS	Least square criterion
Spectral seriation [59]	Spectral	2-Sum criterion
Visual Assessment of Tendency [80]	VAT	Other

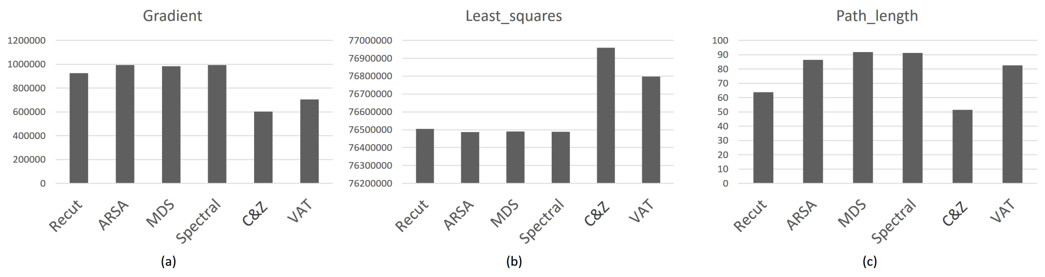

Figure 5.5: Statistical results of comparison between our algorithm and other five previous works: (a) Gradient Measure, (b) Least Square Criterion, and (c) Hamiltonian Path Length

Figure 5.5 shows the comparing result as a bar chart, where "Recut" represents the algorithm in this work, and other abbreviations are listed in Table 5.4 corresponding to the methods. Considering the result optimizing the gradient measure, it can be seen that ARSA, MDS, and Spectral top the performance with nearly no gap between each other, and performance of Recut closely follows the leading tier with a quite small gap. On the other hand, C&Z performs the worst on this criterion, where a significant gap is shown. The situation appears in the case of optimizing the least square criterion, which is shown in Figure 5.5-b.

However, in Figure 5.5-c, C&Z performs the best against the Hamiltonian path length, and the worst performance is provided by ARSA, MDS, and Spectral, which used to be the leading tier on optimizing the previous two criterion. Notably, that performance of Recut follows C&Z, which is still at the second place and significantly better than the rest four algorithms.

We conclude the improvement of our algorithm through this statical result of analysis as follows: performance of Recut is more stable and even. It can be seen that performance of Recut is considerable in all the cases, although it never tops the performance in any case. By contrast, all the outstanding algorithms, which have ever topped the performance against at least one criteria, have at least one weak point. For instance, ARSA performs the best against the gradient measure and the least square criterion, but under performs against the Hamiltonian path length, while C&Z is just in the opposite situation of ARSA.

Visualization of the result permutations also supports our conclusion. Figure 5.6-a,b,c show the permutations generate by ARSA, MDS, and Spectral respectively. It is described clearly by the grey scale in the figures that the matrix rows/columns in these results are sorted continuously and smoothly, and two groups of similar cases of the iris are highlighted clearly.

However, additional information is captured in the permutation by C&Z. In Figure 5.6-d, two bright bands are clearly presented. These bright bands indicate two small groups that are significantly different from both the two big groups. Different from optimizing the gradient measure that is similar to sorting, C&Z generates permutations, where similar rows/columns are located aggregately. As a result, the similarity is highlighted. But still, there is one problem of the permutation generated by C&Z. The two small groups seem to match the same pattern, which means they should belong to one group originally. But they are located separately in the matrix, while they should be merged to one. This is also easy to understand. Permutations optimizing the Hamilton path length do not have to be continuous, also there could be

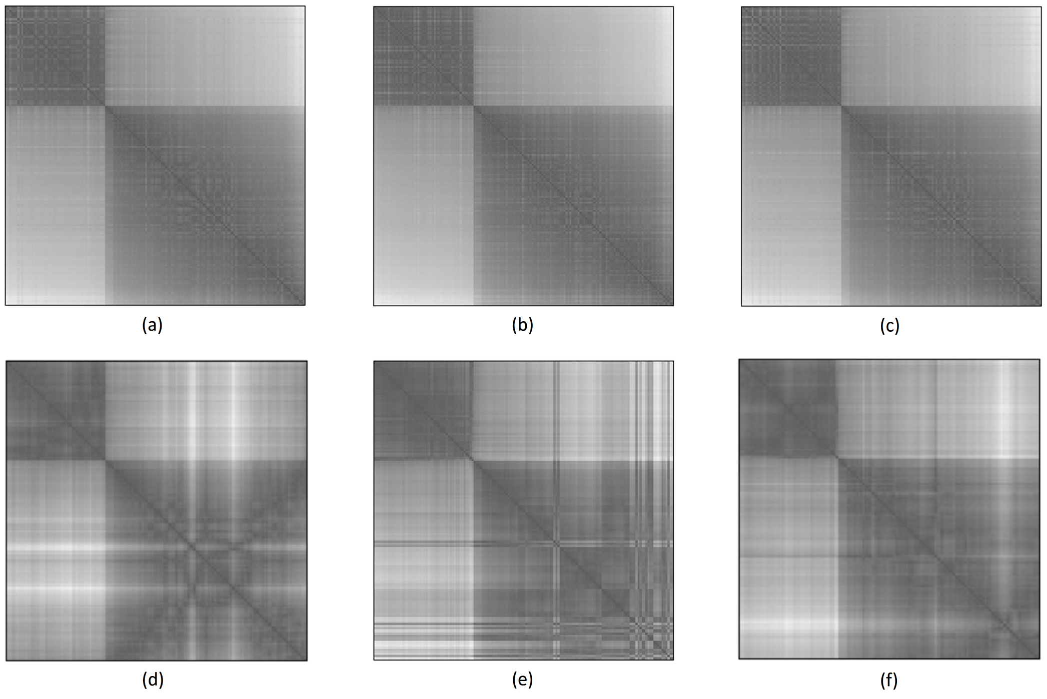

Figure 5.6: Visualization of the permutation generated by both our algorithm and previous works for comparison: (a) ARSA, (b) MDS, (c) Spectral, (d) C&Z, (e) VAT, and (f) Recut

several paths that are all with the shortest path length. Corresponding to the statistical analysis, C&Z under performs in the case of optimizing the gradient measure and the least square criterion. By contrast, Recut provides a permutation satisfying all the requirements. As Figure 5.6-f shows, one bright band with a wider range is presented instead of two in Figure 5.6-d. Also, two big groups are outlined similar to Figure 5.6-a,b,c,d.

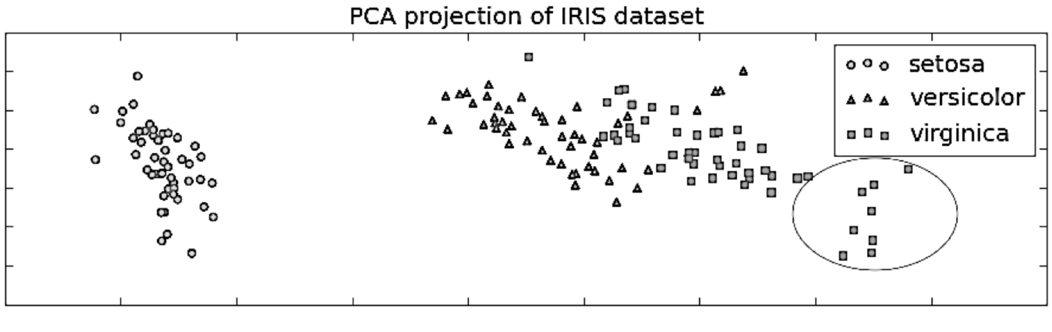

Figure 5.7: PCA projection result of the Fisher dataset, where the outliers are highlighted by a circle

Figure 5.7 shows the PCA result of the original Fisher Iris dataset. Actually three groups can be observed in this figure: the first one on the left side of the figure involving all the Setosa iris cases, the second one on the center-right of the figure with mixed Versicolor and Virginica cases, and the third one on the right-bottom corner with only a few Virginica irises inside (marked by circle in Figure 6). It is clear that the third group is far closer to the second one than to the first one. Corresponding to this PCA result, we can see that the matrix permutation in Figure 5-f is more close to the original dataset. Permutations by ARSA, MDS, and Spectral method are more smooth but contain less information than the rest three ones. On the other hand, although the missing information is captured by C&Z and VAT as well, permutations generated by these two methods are less ordered than that by Recut. In this sense, Recut provides a balanced permutation that considers both the continuous and ordered the appearance of the permutation and the clarity of the matrix structure at the same time.

Based on the observations on permutations and the statical analyzing results, we conclude that Recut can generate the more meaningful result of seriation algorithm by balancing smoothness and aggregation of matrix permutation.

5.2 Evaluation of Visualization

Generally, there are two aspects for evaluating visualization technologies. First is how accurate the visualization describes target data, and second, is how efficient the visualization assists users in further data analysis. Experiments in previous sections aim at evaluating the accuracy of the visual description. In this section, three case studies are conducted for testing the accessibility of our visualization approach from different perspectives.

The first study is for numerically evaluating the dynamic layout of the visualization. With a numerical evaluation, it can be strictly and clearly illustrated in which degree knowledge discovery on the visualization can be assisted. For achieving this goal, we borrow the evaluating metrics from previous works of evaluating the dynamic layout of the treemap and provides both a tracking result overtime and a final metric value in average.

The second study is for learning the readability of the visualization through a human-centered experiment. The most direct criteria come from the user experience, as data visualization is designed for the purpose of human-computer interaction. As a result, subjective evaluation by human being is essential for evaluating data visualization technology. The experiment is based on a recognition procedure on a real dataset of a membership network of several organizations before American Revolution, and a comparison is performed between our visualization and traditional node-link diagrams.

The final study is a preliminary application of the visualization for knowledge discovery. We present the procedure of knowledge discovery on our visualization by following the exploring method introduced in Chapter 4. Dataset employed in this study is the co-purchasing data of books about US policy on Amazon during the 2004 presidential election. By presenting the correlation between online book selling and real policy event as our discovery, I expect to explain and prove the utility of the visualization.

5.2.1 Case Study 1: Layout

The evaluating dynamic layout of data visualization technology is always challenging. There is no benchmark in this domain, as the problem highly depends on both individual subjectiveness and the target data. In this section, we employed two metrics introduced by Bederson and Shneiderman in their work of evaluating treemap layout algorithm [81].

The first criteria is the average aspect ratio, which is defined as follows:

$$AR(t) = \sum_{i=1}^{N_t} max(\frac{w_i}{h_i}, \frac{h_i}{w_i}) \tag{5.4}$$

Where t is the time point of computation, N_t is the number of rectangles appearing in the visualization at time t, w_i is the width of the i^{th} rectangle, and h_i is the height correlated. This value is 1 while all the rectangles appear are perfect squares. Authors of the work [81] stated that squares are more acceptable in visualization for good visibility and convenient labelling. The closer to 1 the average respect ratio is, the better the visual layout may perform.

Another function employed is called the average distance change function. For fitting the situation of our visualization, we adapt the definition of this function as follows:

$$DC(t) = \sum_{i=1}^{N_t} d(i,t) \tag{5.5}$$

Where the distance function $d(i,t)$ is defined like this:

$$d(i,t) = \sqrt{(x_i^t - x_i^{t-1})^2 + (y_i^t - y_i^{t-1})^2 + (w_i^t - w_i^{t-1})^2 + (h_i^t - h_i^{t-1})^2} \tag{5.6}$$

Here, x and y represents the position of the left top corner of the rectangle. In case a rectangle does not exist at time t, we set $x^t = y^t = w^t = h^t = 0$. From the formula, it can be seen that this distance function describes changes in both shape and position of rectangles. If the value is 0, it means there is no change in the layout. The closer to 0 the value is, the more stable the visualization is for users to track on the visualization.

Although, these two metrics are defined for evaluating the dynamic layout of treemaps, we think they are appropriate in our case for two reasons. Firstly, the main presentation form of the visualization in this dissertation is a matrix, where elements inside are either perfect square or rectangle. Second, the position and shape of rectangles can be changed over time in both this work and treemap. In this case, both the average ratio and the average distance metric can be employed for numerically describing how visible and stable the dynamic layout is.

The experiment was conducted on a network generated by the LFR benchmark algorithm by employing the same settings as the previous experiment, especially set $\mu = 0.4$ and network type to be $1000S$. Both of the two metrics were tracked during this process.

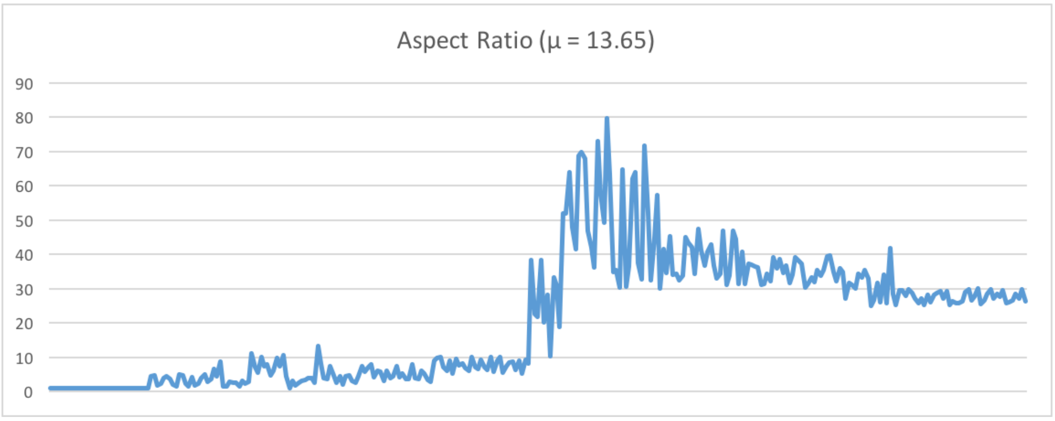

Figure 5.8: Tracking result of the average aspect ratio over time

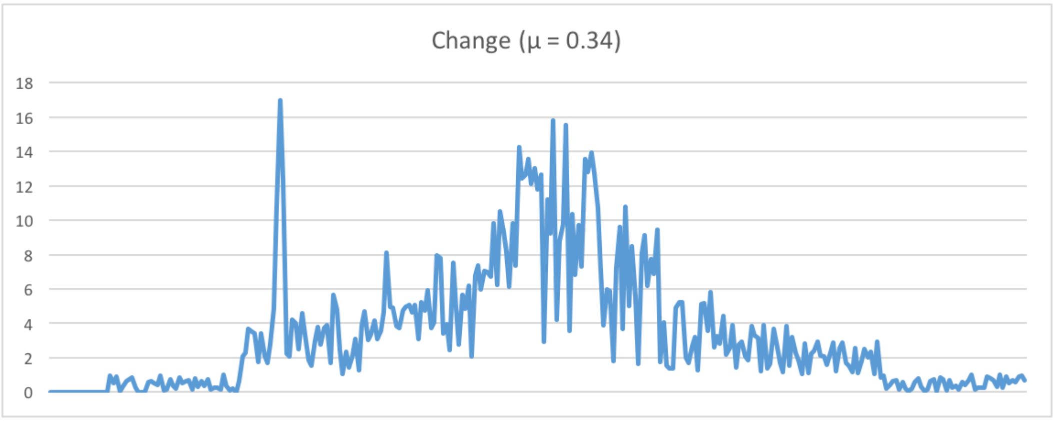

Figure 5.9: Tracking result of the average distance change over time

Figure 5.8 shows a line chart that describes the changes of the average aspect ratio overtime, where the x-axis represents the time passage. It can be seen that the value maintains at a low level of around 5 and fluctuates within a small range during the first half of the whole procedure. Then there is a significant increase in high volatility but finally, recovers to a stable state around 30. In general, this figure shows a stable performance with a short period of shaking. A similar result is observed on the line chart of the average distance change function as shown in Figure 5.9, except for a single peak during the first half of the whole process.

We also computed the average value for both the two functions, where $\mu_{AR} = 13.65$ for the average aspect ratio and $\mu_{DC} = 0.34$ for the average distance change. It is difficult to say how good this performance is. However, we have the experimental result in [81] for evaluating layouts of treemaps as a side explain. The results were built on three experiments with different initial distributions of treemap layout. Results show that if a layout yielded an average distance change less than 0.5, the average aspect ratio was at least 26.10. On the other hand, layout with average aspect ratio around 13 must result in average distance change more than 7. In conclusion, although the layout of this work does not yield a better performance on any of the two metrics singly, it is still substantial by considering the two metrics together.

5.2.2 Case Study 2: Readability

For evaluating the readability, a recognizing process on a real network was simulated on either traditional node-link graph or visualization in this work. The dataset used in this study is a bipartite network describing changes on memberships of 136 people in 5 organizations before American Revolution [1].

The experiment setup is as follows: totally 4 participants were invited, where two of them were university students (notated as S_1, S_2), and another two were employees in the company (notated E_1, E_2). S_1 and S_2 were major in computer science, while E_1 and E_2 were working as soft engineers and product manager respectively. All the four participants were in age between 25 to 35 at the time of the experiment. S_1 and E_2 were males, while S_2 and E_1 were females.

Participants were separated into two groups: group 1 with S_1 as explainer and E_1 as listener, and group 2 with E_2 acting explainer and S_2 as listener. Here, explainer and listener are defined like this: explainer responds to explore the dataset through the node-link diagram and then describes the evolution process to the listener on two aspects: (1) how many communities appears in each time step, and (2) where relation exists between two communities. Listener checks the description from explainer on the visualization in this work and see how much percentage the checking result matches the description.

[1]Dataset can be download from http://konect.uni-koblenz.de/networks/brunson_revolution

Explainers and listeners were trained about how to use their visualization tools separately before the experiment started.

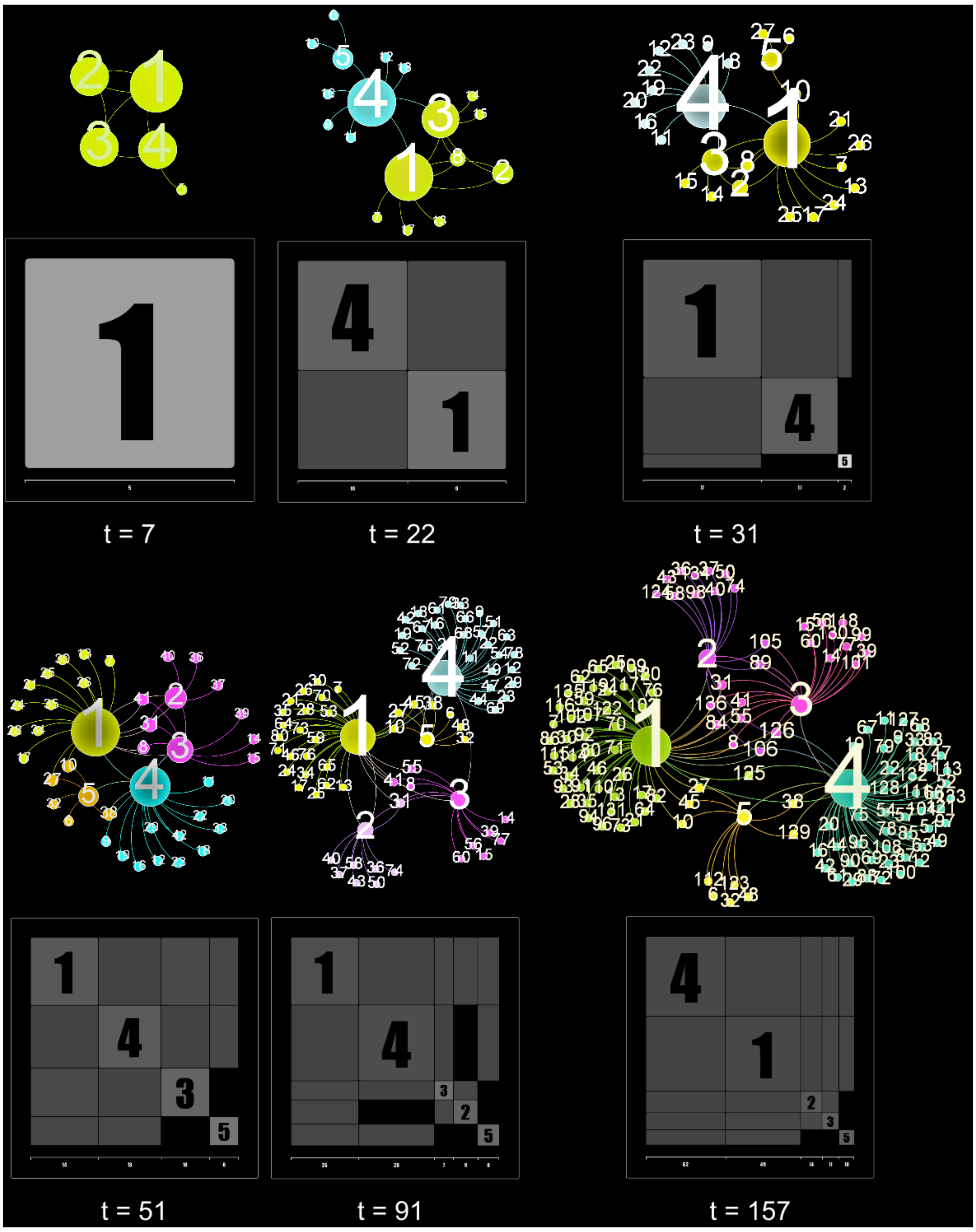

Figure 5.10: Captures of the visualizing result by either our visualization and node-link diagram in case study 2

Figure 5.10 shows six important captures of the visualization by both node-link diagram and approach of this work, where changes of the network community structure appear. The performance of both the two groups was fairly good, more than 95% of the description by explainers were caught by the listener. One divergence of recognition by both group 1 and 2 was at $t = 31$, where community 5 was not recognized as a community by explainers but was captured by listeners. Another interesting observation was at $t = 157$. In group 1, S_1 did not declare that there was a relation between community 2 and 4, while E_1 detected this relation through observing on visualization by this work. Then S_1 checked again on the node-link diagram and finally identified the missed relation.

With this experimental result, it is concluded that approach in this dissertation can provide considerable readability, especially when the network is complex. Details of the evolution process of network community structure can be described accurately, even some information that is not obvious to be observed in node-link diagrams can be clearly detected.

5.2.3 Case Study 3: Exploration

In the final case study for presenting a sample of knowledge discovery, the co-purchasing history of books about US policy during 2004 presidential election was learned as a dynamic network. Background of this story is the competition between George W. Bush, the Republican party candidate and incumbent President, and John Kerry, the Democratic Party candidate. Generally, as common sense, the Republicans used to be the conservative party, while the Democrats used to be the liberal party. Opinions of these two parties are usually antagonistic.

Figure 5.11 shows two captures of the visualization showing the network structure in the different progress of the election, where labels on the elements indicate the indexes of books. Table 5.5 lists detailed information of the books by index.

It was observed that during a long time from the beginning of the election, the co-purchasing behavior only appeared inside the two groups respectively, no matter how these two groups changed. But this situation was broken when

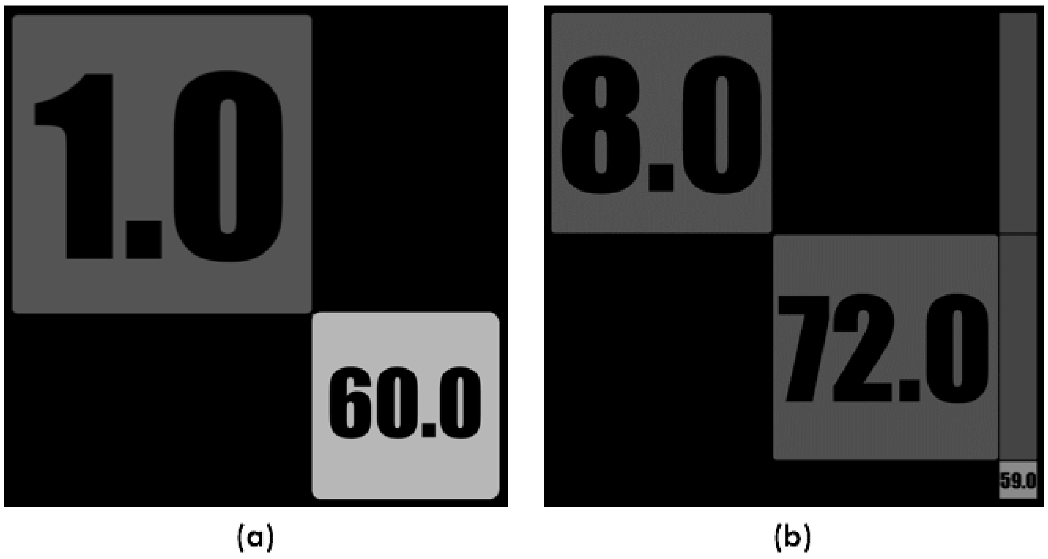

Figure 5.11: Captures of the visualization for two situations of the network: (a) during the most time of the election, and (b) near to the end of the election

Table 5.5: Information of the books appearing in the visualization

Index	**Title**	**Sentiment**
1.0	Bush vs. the Beltway	Conservative
8.0	A National Parity No More	Conservative
60.0	Stupid White Man	Liberal
72.0	Stupid White Man	Liberal
59.0	Why America Slept	Neutral

the election came to the end. A small group containing books with neutral opinions appeared and linked the rest two large groups together, while still no relation shown between the two large groups till the end. This observation perfectly matches the pattern defined in Chapter 4 for detecting potential meaningful discoveries. Combining with the schedule of the presidential election, two observation were concluded to have potential meanings:

- During the period of public voting, co-purchasing is hardly observed between conservative-oriented and liberal-oriented books.
- It may be a good choice to recommend neutral-oriented books to readers preferring either conservative-oriented or liberal-oriented books, and it may be better to avoid the time period of public voting.

By reasoning the observations, it is possible to make discoveries with the potential to be useful, while the exploring process on the visualization is simple and requires little prior knowledge. Discoveries are then expected to effect the decision making, or at least inspire discussion.

Chapter 6

Discussion and Conclusion

6.1 Summary

This dissertation presents a solution of online visualization and exploration of dynamic networks. This approach is designed and implemented according to three levels of data visualization task: automatic processing on the large dataset for building visualization as the lowest level, appropriate design of visual interface as the middle level, and efficient exploring method on the visualization for supporting further data analysis and knowledge discovery at the highest level. Experiments are conducted as evidence for proving the efficiency of our method from different angles. Contributions in this dissertations are summarized as follows:

The lowest level involves two algorithms for dynamically detecting community structure of the network and rearranging the layout of the visualization. The community detection algorithm is designed based on the accurate description of splitting/merging events of communities. Bipartition ratio of community is defined as the splitting/merging condition. With a high efficiency of the algorithm execution, the algorithm can provide stable performance on networks with different complexities. On the other hand, the major presentation of our visualization is the adjacency matrix of the community, where layout changes overtime according to network dynamic. The matrix seriation algorithm in this work solves the dynamic layout problem by recursively cutting and recombining the order of matrix rows/columns for a local optimizing order and searching the global optimization by simulated annealing.

The middle level is designed on the most important concept that is the improvement of efficiency, which is reflected by reducing both the interface cost of computation and the learning cost of users. The visual interface was developed to be 2 dimensional, matrix-like, and animation-based. On the same task of visualizing dynamic networks, It is quicker and cheaper to generate 2-dimensional representation rather than 3 dimensional one. On the other hand, different from the timeline-based approach that may lead to information overload by providing a large number of views, animation-based approach captures the temporal change of the network structure in only one view, and it is easier for users to track entities in the visualization by simply tracking their movements. Furthermore, comparing with a node-link diagram, matrix representation helps reduce the interface cost by involving fewer entities, and simplify the recognition of topological structure, especially for large and dense networks.

The highest level is based on the further usage of the observations from visualization. As data visualization is one part of data analysis, observations through data visualization should serve further analysis and decision making. Meanwhile, an appropriate guidance leads users to quickly focus on knowledge discovery, especially for cases where users have little prior knowledge about the task. The exploring method is inspired by the procedure of chance discovery via KeyGraph. Similar to the bridge-island structure of KeyGraph, we define the pattern of a potential meaningful discovery as a smaller community linking at least two larger communities.

Corresponding to our contributions on the three levels, we conduct several experiments to support our proposition. Parameter settings were learned for both the two underlying algorithms, and algorithm performance was evaluated by comparing with previous outstanding algorithms. Experiments with statistical analysis support that (1) the matrix seriation algorithm in this work performs better on highlighting hidden patterns and information while considering the smoothness and continuance of matrix permutation at the same time, and (2) the community detection algorithm yields a more stable performance than previous works, especially when the boundaries between communities are vague. On the other hand, visual layouts were evaluated through both statistical and empirical experiments. The tracking records of the average aspect ratio and the average distance change show a considerable level of visual accessibility, while considerable readability of the visual interface was proved through a human-centered experiment. Finally, a case study was conducted by showing a sample of knowledge discovery on our visualization. The result highlights the potential of making useful observations through this work.

6.2 Potential Impact and Application

As mentioned in Chapter 1, it is expected that this dissertation can make contributions to real-time analytic on dynamic networks. This section discusses the reason about (1) why visualization technology is essential in real-time analysis, and (2) what the application scenarios expected to be.

Generally speaking, the most challenging problem for real-time analytic systems is the human involved decision making. There is always a contradiction between the human decision and the limit of time. In many cases like buying and selling of stocks, there is generally no time waiting for the human decision. However, it is also unconscionable to trust the data analysis without reservation. At least at this stage, the final decisions are still made by humans. For this reason, an approach to communication between automatic analyzing systems and human beings is essential in case of real-time analytics. Among all the approaches, visualization technology is the most intuitive and simple way for building this communication, where users even do not have prior knowledge of data analysis.

Combining with the tasks of real-time analytics, some application scenarios are proposed in the following paragraphs, in which approaches in this dissertation has a potential to play an important role. However, it is also expected that usage is not limited to the following scenarios, and even not limited to real-time analytics.

Taxi Reservation: The Dispatching Problem

Dispatching problem is quite common in the economy, production, and many other aspects of the daily life. Recently, one of the most popular instances of scheduling problem is transportation dispatching. Many applications on mobile phones, such as Uber, Didi, and Line Taxi, provide the taxi reservation service to public. Then, there is a dispatching problem for optimizing the utilization ratio of taxi resource. This problem comes from the demand of drivers. From the end of drivers, there are generally two rules for scheduling their path, the demand density rule and the random rule. By following the demand density rule, drivers tend to wait at or drive around in ranges with large demand of taxi, and the random rule is just driving with no significant purpose. Both of the two rules are empirical, which usually lead to unbalance of taxi resource and traffic jam. Taxi distribution may concentrate on only a few areas, where traffic jams appear. Meanwhile, the demand of taxi from other places may not be satisfied.

Previous approaches fall into two categories: agent-based simulation and rule-based prediction. The former learns strategies of dispatching by simulating the transportation situation based on a large amount of history data, while the latter predicts the future situation based on rules, such as sequencing rules, flow allowance rules, etc. Predictions can be generated in real time. However, as a real-time system, it definitely faces the human involved decision-making problem. Real-time transportation depending on areas should be provided to drivers with the dispatching recommendations so that drivers can make decisions of driving path.

In this case, we expect visualization in this work to be employed base on the following scenario or similar: visualization is built on the transportation network of taxi, where community elements represent the different areas, size of community elements represent the scale of taxi resource, and density of the relation elements represents the scale of taxi flow between two areas. Then small areas strongly linked to multiple bigger areas are suggested. It is considered that these areas tend to be hidden central areas, to where taxi converge from different sources, and it is expected that large number of passengers may exist in hidden central areas. Taxi resources are then redistributed to avoid blind searching and traffic jams.

Stock Marketing: Prediction of Dynamics

In the past decade, financial networks have attracted much attention from the research community. Stock correlation network is one type of financial network based on the correlation between stock pairs. This correlation could be on stock prices, company relations, social events, etc. Stock correlation network is dynamic because the correlations between stocks change over time. It has been proved that correlation between stocks affect the prediction of stock price, and rules like the average distance rule have bee discovered [82].

Stock buying and selling is a real-time problem based on stock marketing. The most basic rule of Investors to maximize their capital is buying stocks at a low price and selling them at high. However, the problem is nobody can accurately predict the trend of stock price. Given a number of stocks with price decreased in the past few days/weeks/months, the problem is to decide which stock(s) will have price warming up again in the near future.

There is a potential that this dissertation provides a solution. The application scenario could be like this or similar: visualization is built on the stock correlation network, where community elements shows the communities of stocks, size and color of the community elements show the general trend of prices increasing/decreasing (red and large community element shows a significant increasing trend of price inside the community in general, while green and small community element shows slight decreasing trend of price), and connection elements describe the linking strength between communities. The visualization then suggests small green communities linking at least two bigger red communities. We expect that stocks inside the suggested communities are suffering a temporary valley, and they have the potential to meet a bounce as their close neighbors on the network perform sharply increase of price.

6.3 Further Works

This research is not finished by this dissertation, there are still some works remained for further research. Works are listed as follows:

Visualization of inside-community structure Based on the comments from the participants of the human-centered experiment in Chapter 5, the most important improvement for further research is considered to be a detailed visualization of the inner-community structure. One participant in the case study in section 5.2.2 mentioned that only providing the label of a central node in the community is not enough for further analyzing and reasoning the observation, while another participant also suggested that detailed information can enhance the confidence of making observations by users. By agreeing with the suggestions of adding detailed information of inside-community structure, it should also be considered carefully about the way of representation. Several issues exist in this case: the community elements in the visualization can be quite small, where detailed information can only be represented through zooming function, or by a pop-up sub-view responding to mouse events. Also, the nature of online system decides that the view of interface changes rapidly, where expressing detailed information inside the community is quite difficult for easy recognition.

Improvement of visual design based on real application The feasibility of applying this approach to solving real problems also needs to be discussed. In this dissertation, visualization approach is designed on the theoretical level, and some real problems are not considered in details. For example, the data transferring problem is always a challenge in real-time analytics. Data streams are generally unstructured or semi-structure, which need to be processed for analysis. Also, data streams may come from different sources, and parallel strategy of multiple data streams processing is also an issue. In this case, both the visual design and the processing mechanism of data stream should be revised closer to the real case.

Improvement of the matrix seriation algorithm Our method only considers the physical position of the elements in the visualization, while other aspects could affect the result as well, such as the semantic relations of the elements, the social relations if possible, and so on.

Improvement of the visual interface implementation There should be some extension of the visual interface for fitting to detailed purpose. For example, our visualization can only describe a pairwise relation between communities, while there could be relations between more than 2 communities, which cannot be described ideally. In this case, a hierarchical description of a higher-level structure may be useful. Also, it could be difficult to observe the small communities. This issue comes to be tricky and should be solved more smartly.

Bibliography

[1] Lu X, Brelsford C. Network structure and community evolution on twitter: human behavior change in response to the 2011 Japanese earthquake and tsunami. Scientific reports. 2014;4:6773.

[2] Stone J, Developers NT, Eargle J, Sethi A, Li L, Luthey-Schulten Z. Dynamical Network Analysis. 2012;.

[3] Bach B, Pietriga E, Fekete JD. Visualizing dynamic networks with matrix cubes. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM; 2014. p. 877–886.

[4] Herman I, Melançon G, Marshall MS. Graph visualization and navigation in information visualization: A survey. Visualization and Computer Graphics, IEEE Transactions on. 2000;6(1):24–43.

[5] Qi J, Ohsawa Y. Matrix-like visualization based on topic modeling for discovering connections between disjoint disciplines. Intelligent Decision Technologies. 2015;(Preprint):1–11.

[6] Qi J, Ohsawa Y. BLOCKS: Efficient and Stable Online Visualization of Dynamic Network Evolution. The Review of Socionetwork Strategies. 2016;10(1):33–51.

[7] Qi J, Ohsawa Y. Recut: a seriation algorithm balancing smooth display and aggregated features. Fundamenta Informaticae. 2016;146(3).

[8] Ohsawa Y. Chance discoveries for making decisions in complex real world. New Generation Computing. 2002;20(2):143–163.

[9] Fayyad UM, Wierse A, Grinstein GG. Information visualization in data mining and knowledge discovery. Morgan Kaufmann; 2002.

[10] Ghoniem M, Fekete JD, Castagliola P. A comparison of the readability of graphs using node-link and matrix-based representations. In: Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on. Ieee; 2004. p. 17–24.

[11] Misue K, Eades P, Lai W, Sugiyama K. Layout adjustment and the mental map. Journal of visual languages and computing. 1995;6(2):183–210.

[12] Brandes U, Wagner D. A Bayesian paradigm for dynamic graph layout. In: Graph Drawing. Springer; 1997. p. 236–247.

[13] Lee YY, Lin CC, Yen HC. Mental map preserving graph drawing using simulated annealing. In: Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation-Volume 60. Australian Computer Society, Inc.; 2006. p. 179–188.

[14] Burch M, Schmidt B, Weiskopf D. A matrix-based visualization for exploring dynamic compound digraphs. In: 2013 17th International Conference on Information Visualisation. IEEE; 2013. p. 66–73.

[15] Stein K, Wegener R, Schlieder C. Pixel-oriented visualization of change in social networks. In: 2010 International Conference on Advances in Social Networks Analysis and Mining. IEEE; 2010. p. 233–240.

[16] Brandes U, Nick B. Asymmetric relations in longitudinal social networks. Visualization and Computer Graphics, IEEE Transactions on. 2011;17(12):2283–2290.

[17] Doncheva NT, Assenov Y, Domingues FS, Albrecht M. Topological analysis and interactive visualization of biological networks and protein structures. Nature protocols. 2012;7(4):670–685.

[18] Sebrechts MM, Cugini JV, Laskowski SJ, Vasilakis J, Miller MS. Visualization of search results: a comparative evaluation of text, 2D, and 3D interfaces. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM; 1999. p. 3–10.

[19] Greilich M, Burch M, Diehl S. Visualizing the evolution of compound digraphs with TimeArcTrees. In: Computer Graphics Forum. vol. 28. Wiley Online Library; 2009. p. 975–982.

[20] Burch M, Vehlow C, Beck F, Diehl S, Weiskopf D. Parallel edge splatting for scalable dynamic graph visualization. Visualization and Computer Graphics, IEEE Transactions on. 2011;17(12):2344–2353.

[21] Dwyer T, Eades P. Visualising a fund manager flow graph with columns and worms. In: Information Visualisation, 2002. Proceedings. Sixth International Conference on. IEEE; 2002. p. 147–152.

[22] Erten C, Kobourov SG, Le V, Navabi A. Simultaneous graph drawing: Layout algorithms and visualization schemes. In: Graph Drawing. Springer; 2003. p. 437–449.

[23] Federico P, Aigner W, Miksch S, Windhager F, Zenk L. A visual analytics approach to dynamic social networks. In: Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies. ACM; 2011. p. 47.

[24] Itoh M, Toyoda M, Kitsuregawa M. An interactive visualization framework for time-series of web graphs in a 3D environment. In: Information Visualisation (IV), 2010 14th International Conference. IEEE; 2010. p. 54–60.

[25] Boyandin I, Bertini E, Lalanne D. A Qualitative Study on the Exploration of Temporal Changes in Flow Maps with Animation and Small-Multiples. In: Computer Graphics Forum. vol. 31. Wiley Online Library; 2012. p. 1005–1014.

[26] Diehl S. Preserving the Mental Map using Foresighted Layout Stephan Diehl, Carsten Giorg and Andreas Kerren University of Saarland, FR 6.2 Informatik, PO Box 15 11 50, D-66041 Saarbriucken, Germany. 2001;.

[27] Diehl S, Görg C. Graphs, they are changing. In: Graph drawing. Springer; 2002. p. 23–31.

[28] Erten C, Harding PJ, Kobourov SG, Wampler K, Yee G. GraphAEL: Graph animations with evolving layouts. In: Graph Drawing. Springer; 2003. p. 98–110.

[29] Ohsawa Y, Ito T, Kamata MI. Kamishibai KeyGraph: Tool for Visualizing Structural Transitions for Detecting Transient Causes. New Mathematics and Natural Computation. 2010;6(02):177–191.

[30] Sugimoto M, Ueda T, Okada S, Ohsawa Y, Maeno Y, Nitta K. Discussion analysis using temporal data crystallization. In: New Frontiers in Artificial Intelligence. Springer; 2012. p. 205–216.

[31] Vehlow C, Beck F, Auwärter P, Weiskopf D. Visualizing the evolution of communities in dynamic graphs. In: Computer Graphics Forum. vol. 34. Wiley Online Library; 2015. p. 277–288.

[32] Ma C, Kenyon RV, Forbes AG, Berger-Wolf T, Slater BJ, Llano DA. Visualizing dynamic brain networks using an animated dual-representation. In: Proceedings of the Eurographics Conference on Visualization (EuroVis); 2015. p. 73–77.

[33] Mall R, Langone R, Suykens JA. Netgram: Visualizing Communities in Evolving Networks. PloS one. 2015;10(9):e0137502.

[34] Girvan M, Newman ME. Community structure in social and biological networks. Proceedings of the national academy of sciences. 2002;99(12):7821–7826.

[35] Clauset A, Newman ME, Moore C. Finding community structure in very large networks. Physical review E. 2004;70(6):066111.

[36] Guimera R, Sales-Pardo M, Amaral LAN. Modularity from fluctuations in random graphs and complex networks. Physical Review E. 2004;70(2):025101.

[37] Massen CP, Doye JP. Identifying communities within energy landscapes. Physical Review E. 2005;71(4):046101.

[38] Medus A, Acuna G, Dorso C. Detection of community structures in networks via global optimization. Physica A: Statistical Mechanics and its Applications. 2005;358(2):593–604.

[39] Guimera R, Amaral LAN. Functional cartography of complex metabolic networks. Nature. 2005;433(7028):895–900.

[40] Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment. 2008;2008(10):P10008.

[41] Donetti L, et al. Detecting network communities: a new systematic and efficient algorithm. Journal of Statistical Mechanics: Theory and Experiment. 2004;2004(10):P10012.

[42] Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D. Defining and identifying communities in networks. Proceedings of the National Academy of Sciences of the United States of America. 2004;101(9):2658–2663.

[43] Newman ME, Leicht EA. Mixture models and exploratory analysis in networks. Proceedings of the National Academy of Sciences. 2007;104(23):9564–9569.

[44] Ronhovde P, Nussinov Z. Multiresolution community detection for megascale networks by information-based replica correlations. Physical Review E. 2009;80(1):016109.

[45] Rosvall M, Bergstrom C. Maps of information flow reveal community structure in complex networks. Citeseer; 2007.

[46] Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences. 2008;105(4):1118–1123.

[47] Newman ME, Girvan M. Finding and evaluating community structure in networks. Physical review E. 2004;69(2):026113.

[48] Palla G, Derényi I, Farkas I, Vicsek T. Uncovering the overlapping community structure of complex networks in nature and society. Nature. 2005;435(7043):814–818.

[49] Enright AJ, Van Dongen S, Ouzounis CA. An efficient algorithm for large-scale detection of protein families. Nucleic acids research. 2002;30(7):1575–1584.

[50] Fortunato S, Barthelemy M. Resolution limit in community detection. Proceedings of the National Academy of Sciences. 2007;104(1):36–41.

[51] Hubert L, Arabie P, Meulman J. Combinatorial data analysis: Optimization by dynamic programming. vol. 6. SIAM; 2001.

[52] Chen CH. Generalized association plots: Information visualization via iteratively generated correlation matrices. Statistica Sinica. 2002;p. 7–29.

[53] Brusco MJ, Stahl S. Branch-and-bound applications in combinatorial data analysis. Springer Science & Business Media; 2006.

[54] Brusco MJ, Köhn HF, Stahl S. Heuristic implementation of dynamic programming for matrix permutation problems in combinatorial data analysis. Psychometrika. 2008;73(3):503–522.

[55] Caraux G, Pinloche S. PermutMatrix: a graphical environment to arrange gene expression profiles in optimal linear order. Bioinformatics. 2005;21(7):1280–1281.

[56] Hubert L, Schultz J. Quadratic assignment as a general data analysis strategy. British journal of mathematical and statistical psychology. 1976;29(2):190–241.

[57] Barnard ST, Pothen A, Simon H. A spectral algorithm for envelope reduction of sparse matrices. Numerical linear algebra with applications. 1995;2(4):317–334.

[58] Rodgers JL, Thompson TD. Seriation and multidimensional scaling: A data analysis approach to scaling asymmetric proximity matrices. Applied psychological measurement. 1992;16(2):105–117.

[59] Ding C, He X. Linearized cluster assignment via spectral ordering. In: Proceedings of the twenty-first international conference on Machine learning. ACM; 2004. p. 30.

[60] Gruvaeus G, Wainer H. TWO ADDITIONS TO HIERARCHICAL CLUSTER ANALYSIS†. British Journal of Mathematical and Statistical Psychology. 1972;25(2):200–206.

[61] Climer S, Zhang W. Rearrangement clustering: Pitfalls, remedies, and applications. The Journal of Machine Learning Research. 2006;7:919–943.

[62] McCormick Jr WT, Schweitzer PJ, White TW. Problem decomposition and data reorganization by a clustering technique. Operations Research. 1972;20(5):993–1009.

[63] Niermann S. Optimizing the ordering of tables with evolutionary computation. The American Statistician. 2012;.

[64] Lenstra J. Technical Note—Clustering a Data Array and the Traveling-Salesman Problem. Operations Research. 1974;22(2):413–414.

[65] Wittek P. Two-way incremental seriation in the temporal domain with three-dimensional visualization: Making sense of evolving high-dimensional datasets. Computational Statistics & Data Analysis. 2013;66:193–201.

[66] McBurney P, Ohsawa Y. Chance Discovery. 2003;.

[67] Ohsawa Y, Benson NE, Yachida M. KeyGraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In: Research and Technology Advances in Digital Libraries, 1998. ADL 98. Proceedings. IEEE International Forum on. IEEE; 1998. p. 12–18.

[68] Ohsawa Y. KeyGraph: visualized structure among event clusters. In: Chance Discovery. Springer; 2003. p. 262–275.

[69] Ohsawa Y. KeyGraph as risk explorer in earthquake–sequence. Journal of contingencies and crisis management. 2002;10(3):119–128.

[70] Horie K, Ohsawa Y, Okazaki N. Products designed on scenario maps using pictorial KeyGraph. WSEAS Transactions on Information Science and Applications. 2006;3(7):1324–1331.

[71] Seo Y, Iwase Y, Takama Y. KeyGraph-based BBS for online chance discovery. In: Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on. vol. 2. IEEE; 2006. p. 1754–1758.

[72] Goldberg DE, Sastry K, Ohsawa Y. Discovering deep building blocks for competent genetic algorithms using chance discovery via keygraphs. In: Chance discovery. Springer; 2003. p. 276–301.

[73] Liu C, Ohsawa Y, Suda Y. Valuation of data through use-scenarios in innovators' marketplace on data jackets. In: 2013 IEEE 13th International Conference on Data Mining Workshops. IEEE; 2013. p. 694–701.

[74] Ohsawa Y, Liu C, Suda Y, Kido H. Innovators marketplace on data jackets for externalizing the value of data via stakeholders' requirement communication. In: 2014 AAAI Spring Symposium Series; 2014. .

[75] Strehl A, Ghosh J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. The Journal of Machine Learning Research. 2003;3:583–617.

[76] Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Physical review E. 2008;78(4):046110.

[77] Lancichinetti A, Fortunato S. Community detection algorithms: a comparative analysis. Physical review E. 2009;80(5):056117.

[78] Fisher RA. The use of multiple measurements in taxonomic problems. Annals of eugenics. 1936;7(2):179–188.

[79] Buchta C, Hornik K, Hahsler M. Getting things in order: an introduction to the R package seriation. Journal of Statistical Software. 2008;25(3):1–34.

[80] Bezdek JC, Hathaway RJ. VAT: A tool for visual assessment of (cluster) tendency. In: Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on. vol. 3. IEEE; 2002. p. 2225–2230.

[81] Bederson BB, Shneiderman B, Wattenberg M. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. AcM Transactions on Graphics (TOG). 2002;21(4):833–854.

[82] Onnela JP, Chakraborti A, Kaski K, Kertesz J. Dynamic asset trees and Black Monday. Physica A: Statistical Mechanics and its Applications. 2003;324(1):247–252.