A Hybrid Drawing Style for Semi—Bipartite Graphs

Qi Zhou

(Master's Program in Computer Science)

Advised by Kazuo Misue

Submitted to the Graduate School of
Systems and Information Engineering
in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering
at the
University of Tsukuba

March 2011

Abstract

Semi–bipartite graphs — bipartite graphs with edges within one partition — are a kind of graph that can be found in various fields of real life. Visualizing of such graph is thought to be valuable but this kind of graph has not attracted much attention until recently.

Regarding the purpose of visualizing semi–bipartite graphs with high readability, we present a drawing style combined with the anchored map—a drawing method for bipartite graph, and matrix representation—one of the most traditional representation methods.

In order to drawing semi–bipartite graphs in this hybrid drawing method, first we extended the drawing method anchored map to be able to handle semi–bipartite graphs, in order to further improve the readability and support data mining, we tried to introduce matrix representation into the anchored map, and for display matrices with good readability we developed a matrix reordering algorithm based on barycenter heuristic.

To evaluate our hybrid drawing method, first we do an evaluation experiment on both the extended anchored map and the matrix reordering algorithm. Then we will show the characters of our method by showing the same drawing result of real data with a semi–bipartite graph structure.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Information Visualization and Graph Drawing

Information visualization is a set of technologies that produces visual representations of abstract data, and its purpose is to amplify cognitive performance to reinforce human cognition, enabling the viewer to gain knowledge about the internal structure of the data and causal relationships in it.

A graph is an abstract structure that is used to model information. Graphs may be used to represent any information that can be modeled as objects (node) and connections between those objects (edge). Unfortunately graphs with large information are always hard to read and understood, so how to draw graphs automatically with good readability becomes a problem. In this paper, we are focusing on the drawing a kind of graph called "semi−bipartite graph" — bipartite graphs with edges within one partition.

1.2 Bipartite Graph and Semi−bipartite Graph

A bipartite graph is a graph whose nodes can be divided into two disjoint sets A and B such that every edge connects a node in A to one in B, that is, A and B are independent sets. A bipartite graph can be formally described as $G = (A \cup B, E)$. E is a finite set of edges, and $E \subseteq A \times B$. Bipartite graphs are a kind of graph which has been well studied

Semi−bipartite graphs can be defined as $G = (A \cup B, E_1 \cup E_2)$, where A and B are two finite sets of nodes, E_1 is a finite set of edges between A and B, i.e., $E_1 \subseteq \{(u,v) \mid u \in A, v \in B\}$,and E_2 is a finite set of edges between the nodes in B, i.e., $E_2 \subseteq \{(u,v) \mid u, v \in B\}$ or $E_2 \subseteq \{\{u,v\} \mid u, v \in B\}$. In our research E_2 can be both direct and undirected. And the only difference between bipartite graph and semi−bipartite graph is the existence of E_2 edges which exist inside node set B.

1.3 Drawing of Semi−bipartite Graph

There are two kinds of nodes and edges that exist in a semi−bipartite graph, how to visualize those two kinds of nodes as well as their two different relations in good readability at the same time is a challenging work. Especially when dealing with large scale data.

The structure of the semi−bipartite graph has not brought much attention until recently. The model of semi−bipartite graph has been recently introduced by Xu et al. [17], he also argued the importance of visualizing semi−bipartite graph, and proposed three layout algorithms for visualizing gene ontology networks (semi − bipartite graph with hierarchical structure).

1.4 Purpose and Approach

The purpose of our research is to draw semi−bipartite graphs to reveal their two kinds of different relations in good readability and support data mining.

To achieve this, two drawing method are proposed. First we extended the drawing method anchored map to be able to draw semi−bipartite graphs where nodes of set A are arranged on a circumference with same interval to decide the global structure of the graph. And in order to further improve the readability and support data mining, we proposed a hybrid drawing style combined with anchored map and matrix representation where nodes of set B are clustered and visualized by matrix representation.

1.5 The Organization of this Paper

In this chapter, we introduced the structure of semi−bipartite graphs and the purpose as well as the approach of our research. In the next chapter, related work such as anchored map and matrix related researches will be introduced. In Chapter 3, we will discuss the characters as well as the requirements for drawing semi−bipartite graphs.

In Chapter 4 we will introduce our proposed drawing style. Then the details of drawing style will be discussed; the extended anchored map for drawing semi−bipartite graphs will be explained in Chapter 5 and in Chapter 6, a matrix reordering algorithm developed for better readability of matrices will be introduced.

In order to evaluate our research, two evaluation experiments are held separately, one is for extended anchored map, and the other is for the proposed matrix reordering algorithm. The characters of two proposed drawing method will be discussed by showing some drawing result of real data in Chapter 7. And lastly the conclusion of our research will be presented.

Chapter 2

Related Work

In this chapter, we show some related works about our research such as graph drawing anchored map, the matrix representation and reordering.

2.1 Graph Drawing

Graph drawing is a field of research with a long history and automatic graph drawing has many important applications in real life such as software engineering, database and web design, networking, and visual interfaces for many other domains [3][11]. However, almost all research of graph drawing is based on either node–link diagrams or adjacency matrix representations (Figure 1).

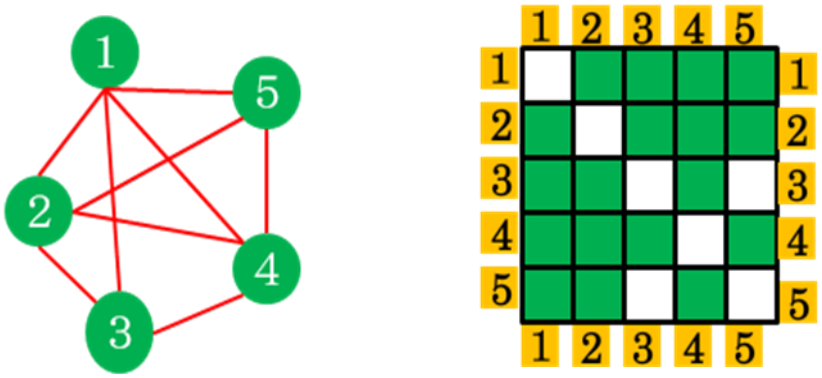


Figure 1 node—link diagram (left) and matrix representation (right)

2.1.1 Node–link diagrams

Node–link diagrams are the most common representation of graphs where nodes are represented by dots and arcs represent the edges between connected nodes. In the graph drawing community, many researches are dealing with layout techniques to satisfy aesthetic criteria such as minimizing edge–crossings, the ratio between the longest edge and the shortest edge, and revealing symmetries [5].

Force–directed is one of the most common ways for drawing node–link diagrams, and one of the earliest heuristics of force–directed placement was based on the spring embedder model[6] where nodes are considered as mutually repulsive charges and edges as springs that attract connected nodes.

Node–link is the most common way of representing graphs but density has a strong impact on readability in these diagrams [10], they become unreadable when visualizing a dense graph. Focusing on basic readability tasks such as finding an actor or determining if two actors are linked, ghoniem et al. found that node–link diagrams perform badly for dense graphs even with few nodes. Spring embedder works in iterations, and in each iteration, the forces exerted on each node v are computed.

2.1.2 Matrix Representation

Bertin first introduced visual matrices to represent graphs in "Semiology of graphics" [2]. Ghoniem et al. [10] showed that matrices representation is better than node−link diagrams when visualizing large graphs or dense graphs in several low—level reading tasks, except path finding. Bertin also qualified matrices as "reorderable" and showed that matrices can be used to display high–level structures (or good readability) by finding good permutations of their rows and columns. Reordering rows and columns of an adjacency matrix is similar to computing the layout for a node–link diagram: finding a layout that satisfies certain aesthetic criteria. Reordering of matrices can be divided into two categories: automatic and interactive.

Automatic reordering for matrices is a well known problem which can be seen in various research areas such as mathematics and biology. Matrix ordering algorithms are always tried to optimize certain objective function (or aesthetic criteria). For example, diagonalizing the matrix, a goal expressed by Bertin, is proved to be a NP–complete problem, but Siirtola and Makinen developed a set of heuristics [26] to find an approximate solution. Spectral methods has been widely used for reordering binary matrices for image compression or DNA sequencing [1]. In our research, we first proposed aesthetic criteria for our matrix representation, and then we developed a matrix reordering algorithm based on barycenter heuristic[22].

In interactive tools such as InfoZoom [27] or TableLens[25], user can quickly identify correlated columns by reordering one dimension of the table according to one attribute (one column). To sort a matrix according to the names, then dates, then category, the user has to order first by category, then by dates and finally, by names which is a long and tedious work.

2.1.3 NodeTrix

In the purpose of visualizing large social networks, Henry et al. [12] presented a hybrid representation that combined the advantages of two traditional representations: node–link diagram and matrix representation. Node–link diagrams are used to show the global structure of a network, while arbitrary portions of the network can be shown as adjacency matrices to better support the analysis of communities. And they also developed a set of interaction techniques allowing the user to create a NodeTrix visualization by dragging selections to and from node—link and matrix forms.

2.2 Drawing of Bipartite Graph

Drawing of bipartite graph is a well researched field where a lot of work has been done. Misue [19][20] proposed a drawing method called “anchored map” (Figure 2), an advanced form of spring embedder model[6], is a drawing technique for visualizing large−scale bipartite graphs. Nodes in one set are called “anchors” which are arranged on a circumference with the same interval, and nodes in another set called “free nodes” are arranged at suitable positions in relation to adjacent anchors by spring embedding. To improve the readability of anchored maps, anchors are arranged so as to reduce edge crossings and edge length. The algorithm to decide the order of anchors for achieving less edge crossing and shorter edge length is proposed by Misue. Base on the anchored map, Ito et al. [14] developed a 3D bipartite graph visualization technique and a drawing method for clustered bipartite graphs [15].

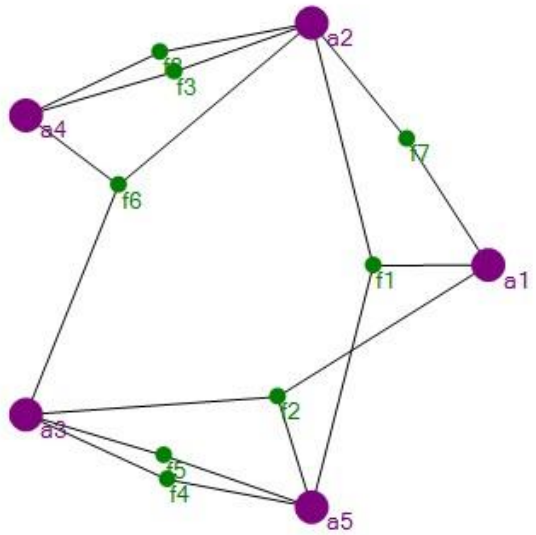


Figure 2 example of anchored map style

Other researches about two−sided bipartite graph drawing or extended models are also exist, for example, Zheng et al. [31] proposed two layout models for bipartite graphs and proved several theorems of edge crossing for these models. Newton et al. [21] proposed new heuristics for two−sided bipartite graph drawing. Giacomo et al. [9] developed a method that drawing bipartite graphs on two curves to avoid edge crossing.

Although the structure of bipartite graph and semi−bipartite graph are close to each other, drawing method for bipartite graph cannot be used for semi−bipartite graph directly especially in two−sided style since edges exist in one set of the nodes. In this paper, we extended anchored map to visualizing semi−bipartite graph.

2.3 Drawing of Semi−bipartite Graph

Although semi−bipartite graph is similar to bipartite graph, researches on visualizing

semi−bipartite graph are relatively less. The model of semi−bipartite graph has been introduced by Xu et al. [17], and he proposed three layout algorithms for visualizing semi −bipartite graph with hierarchical structure which exist in gene ontology networks. Before Xu's definition, the semi−bipartite graph was be called "Multiple—Category Graphs" [16], and Itoh et al. have presented a hybrid method combined with space−filling and force−directed layout for visualizing it, however the node cluster is drawn as a single node so the information(E_2 edges) inside cannot not be read. We have proposed a drawing method combined with the anchored map and the matrix representation for semi-bipartite graph, but algorithms for better layout or readability are not finished at that time [32].

There are also a lot of researches that have been studied on visualizing data with semi−bipartite graph structure, such as social network and papers−authors data. For social networks, researchers always focus on the actor−actor relations [4], community−actor relations can be seen during the related operation by the user instead of visualized on the drawing result directly [18]. And for papers−authors data, the paper citation and co−authorship relations are always visualized separately [7], which means the two relationships (edges) of semi−bipartite graph are not visualized simultaneously.

In the field of graph drawing, to our knowledge, research focusing on visualizing data with semi−bipartite graph without hierarchical structure does not exist yet.

Chapter 3

Drawing of Semi–bipartite graph

In this chapter, we will show the characters of semi–bipartite graph, based on that, we will discuss the demand and requirements of drawing such a graph.

3.1 Characters of Semi–bipartite Graph

As defined in chapter 1.2, Semi–bipartite graphs(Figure 3) can be defined as $G = (A \cup B, E_1 \cup E_2)$. It is obvious that the most important character of semi–bipartite is the existence of two different kinds of nodes and edges. So in order to distinguish its two different nodes and their relationships (edges), semi–bipartite is not suitable for ordinary node–link representation or matrix representation. Xu et al. [17], proposed three layout algorithms for semi – bipartite graph with only hierarchical structure, however hierarchical structure is not a character of semi–bipartite graph.

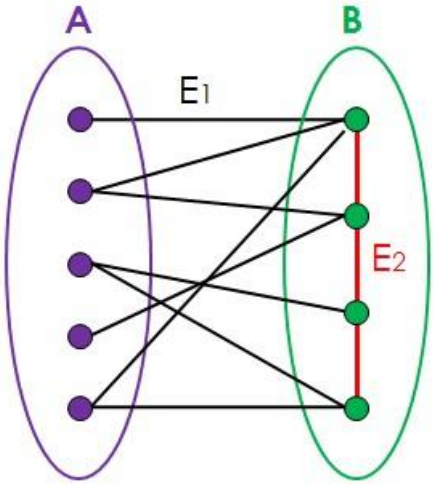


Figure 3 semi–bipartite example

3.2 The Demand of Drawing Semi–bipartite Graph

Semi–bipartite graph have not drawn much attention until recently, but it is a kind graph structure which can be seen in lot of fields such as social networks, author–paper data, and gene ontology networks [17], et al.

A social network is a social structure made up of individuals (or organizations) called

"nodes", which are connected by one or more specific types of interdependency, such as friendship, kinship, common interest or some other relationships. There exists research and systems that either focus on the relationships between people [12][13] or relationships between people and their community. However for real social network service data, although both people–people and people–community relationships exist, and they are considered to have great influence on each other, for example, people belongs the same community are prone to have relationships, researches on visualizing such graph with its two kinds of nodes and edges does not exist, so drawing the social network with its two kinds of relationships is thought to be needed and valuable.

Another example is the author–paper data. Both citation relationship (paper–paper) and co–authorship exist. The existing researches or systems always focuses on the either citation relationship or co–authorship [18]. However, it is considered that papers written by the same author are prone to have citation relations, the possibility of citation relation between papers written by authors studying in different areas is relatively low.

Above all, the semi–bipartite graph exists a lot in the field of real life, and we believe that drawing semi–bipartite graphs with its two kinds of nodes and edges simultaneously is valuable which will be shown in Chapter 7 with some real drawing examples.

3.3 Requirements of Drawing Semi–bipartite Graph

As discussed in previous part, in order to drawing semi–bipartite graph in good readability, there are several requirements as follows:

Req.1 the two different nodes should be easily distinguished (for example, user should quickly understand which one represents people and which one represents communities)
Req.2 the two different edges should be easily distinguished (for example, user should be able to focus on reading one kind of relationship)
Req.3 nodes with close relationships should be placed near to each other (common aesthetic criteria in graph drawing which is also defined as minimizing edge length)

In order to satisfy those three requirements, we developed a hybrid drawing style combined with anchored map and matrix representation, which will be introduced in the next chapter.

Chapter 4

Hybrid Drawing Style Combined with Anchored Map and Matrix Representation

In this chapter, we will generally explain our proposed drawing style. There are mainly 5 steps. The extended anchored map and the matrix representation part are the two main points of our research which will be explained in later 2 chapters.

4.1 Extended Anchored Map for Semi−bipartite Graph

First, in order to satisfy Req.1 in 3.3, we decided to draw a semi−bipartite graph in anchored map style in which nodes of set A are fixed on a circumference as anchors to keep the overview of graph, and nodes of set B are arranged as free nodes by the spring embedding model [6] which will automatically decide their position by the relationship of E_1 and E_2 edges which are drawn in different colors (Figure 4).

In order to satisfy Req.3 in 3.3 , anchors with close relationship should be place near to each other (free nodes with close relationship are automatically arranged by spring embedding model). The anchored map has developed techniques to decide a good anchor order to satisfy Req.3 for bipartite graph, but for semi−bipartite, it cannot be performed directly since the existence of E_2 edges, as seen in Figure 4 left, not only anchors with common free nodes, but also anchors get connected by E_2 edges be close to each other.

In order to find a good anchor order for semi−bipartite graphs, we have extended anchored map by redefining the “penalty” − an index for goodness of drawing result, which will be discussed later.

But there is a problem with drawing a semi−bipartite graph in the anchored map style. When drawing a dense graph which has many edges (especially E_2 edge), the readability can be not assured whenever the anchor order chances, in another words, it cannot be visualized well by the anchored map style (Figure 5). Sato et al. [28] proposed a method based on node clustering (on free node) for bipartite graph, and it is proved that the readability of dense graph or large scale graph can be improved after performing clustering. However, for the semi−bipartite graph, if several free nodes are clustered into one node cluster and represented as a single node, the relationship (E_2 edge) between free

nodes will be lost. In order to solve this problem, we tried to introduce matrix representation into anchored map style (Figure 6). First we perform node clustering on free nodes, and then we use matrices to represent node clusters.

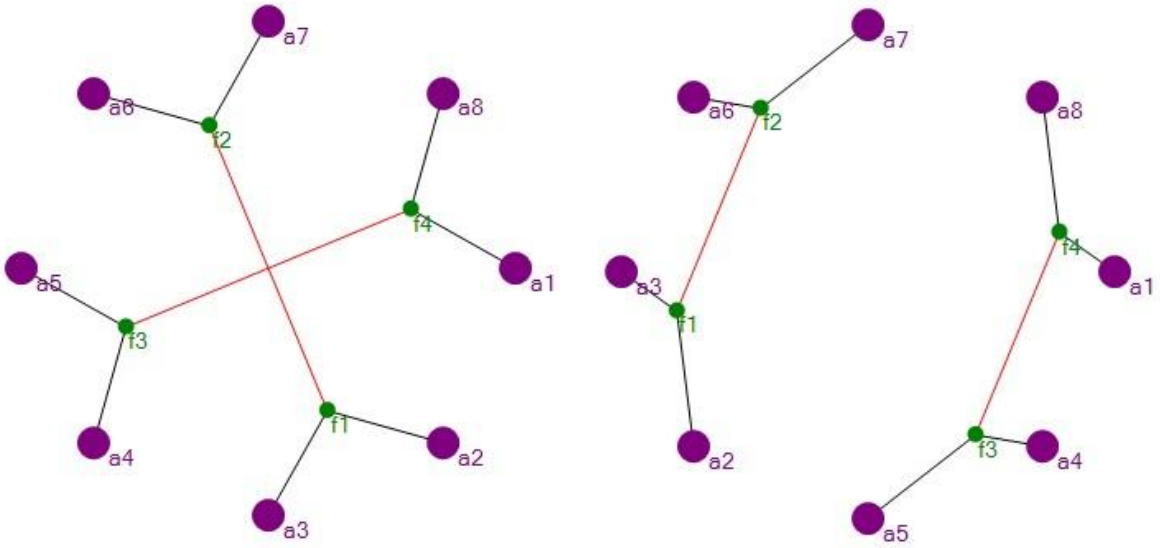


Figure 4 same data with different anchor order

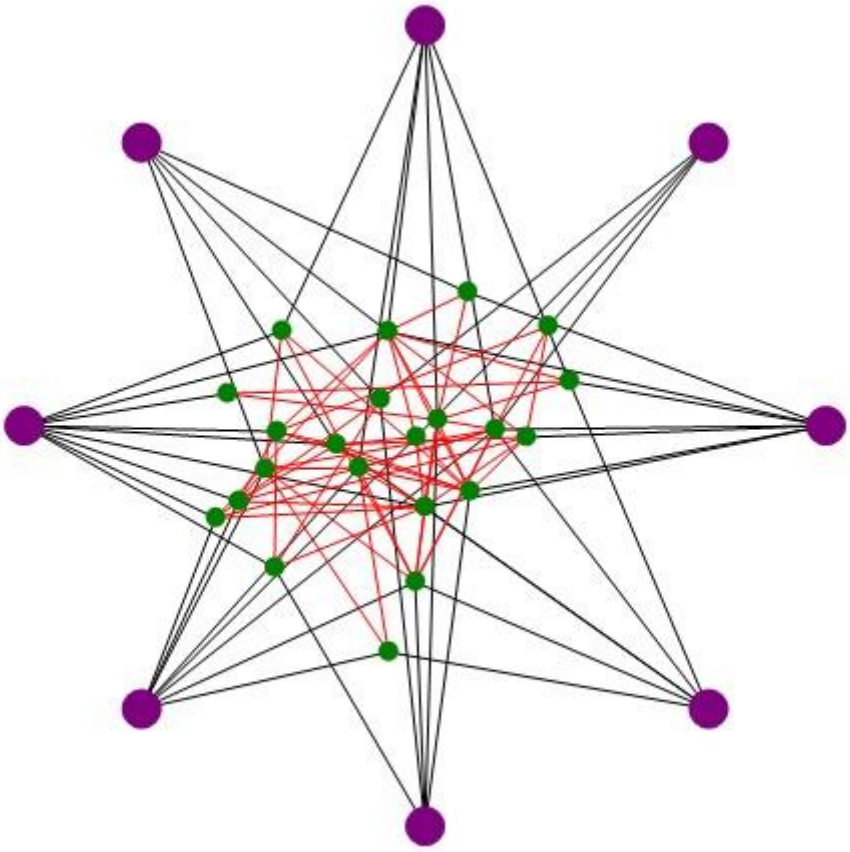


Figure 5 a dense graph drawn in anchored map style

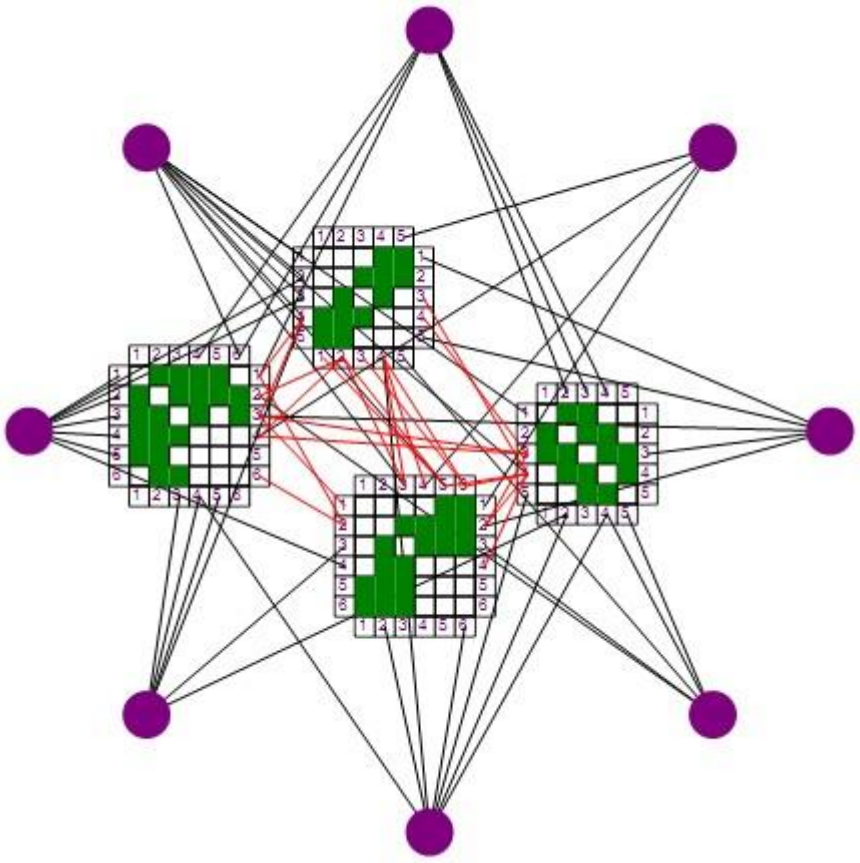


Figure 6 the same graph drawn in hybrid drawing style

4.2 Drawing Object

As discussed in former section, we tried to combine anchored map style with matrix representation. In anchored map style, nodes of set A are fixed on a circumference to decide the overview of graph, then we perform node clustering on nodes of set B (free nodes) and represent the relationship between free nodes inside by matrix style (Figure 1). And after clustering, the structure of semi−bipartite graph will change. Even the possibility of changing into a bipartite graph exists. The definition of the drawing object is defined as follows:

Before： $G = (A \cup B, E_1 \cup E_2)$
After: $G = (A \cup B', E_1' \cup E_2')$

In order to satisfy Req. 2, E_2 should be displayed inside matrices as much as possible, so we decided to perform clustering on nodes of set B based on E_2 edge. There are multiple clustering methods with different characters available, and of course different clustering methods will bring different drawing result. However, the drawing object is the same, and our research is based on drawing such objects in good readability.

After clustering, free nodes connected to each other may be clustered into node clusters,

and at the same time a single node may also exist after clustering (Figure 7). In the proposed drawing style, both node clusters and single nodes are called free nodes, but only node clusters will be drawn in matrices.

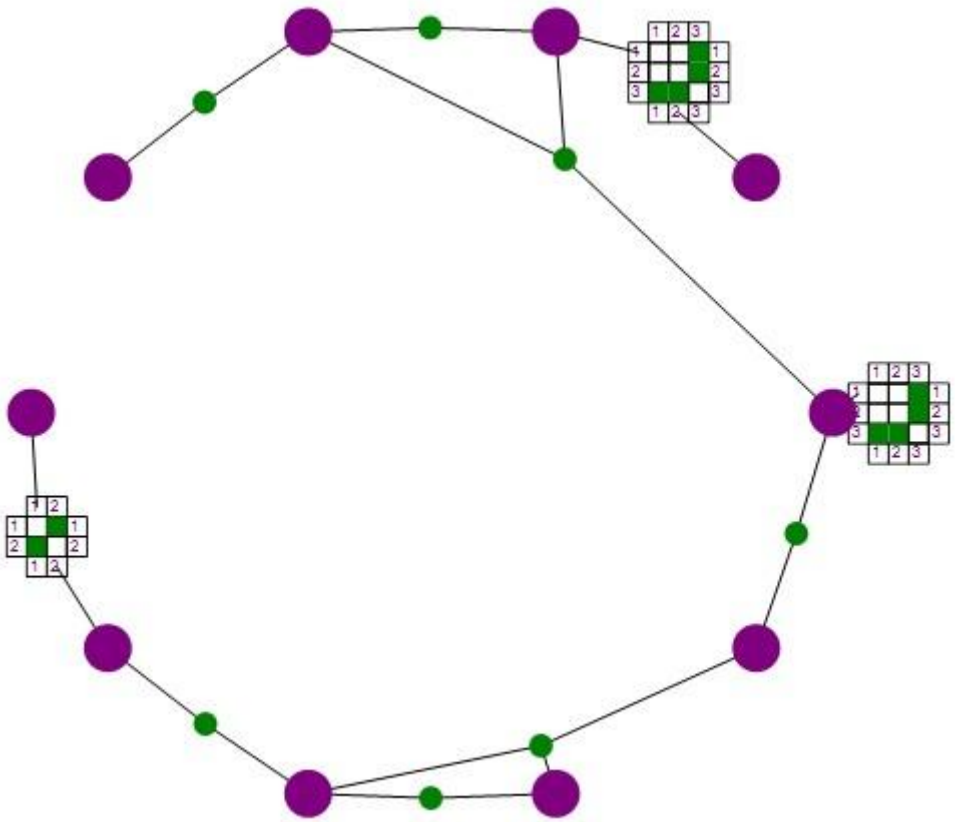


Figure 7 a semi–bipartite graph where node clusters and signle nodes both exist and all E2 edges are inside node clusters(matrices)

4.3 Aesthetic Criteria

We employed the following aesthetic criteria for proposed hybrid drawing style:

(R1) Anchors with close relations are laid out as closely as possible.
(R2) Free nodes connected to common anchors are laid out as closely as possible.
(R3) Free nodes connected to each other are laid out as closely as possible.
(R4) Minimize the total length of edges
(R5) Minimize the number of edges–crossings.
(R6) Free nodes (within single matrix) with close relations are placed near each other as close as possible.

(R4) and (R5) are the two most common aesthetic criteria in the graph drawing area, (R2) and (R3) can be satisfied by using spring embedder model. In this research, we are mainly focusing on (R1) and (R6), details about how (R1) and (R6) are to be satisfied will be discussed in later chapters.

4.4 Procedures

There are mainly 5 steps in our research.

(1) Node clustering on nodes of set B
(2) Fixing nodes of set A on a circumference (extended anchored map)
(3) Layout (Spring Embedding model)
(4) Displaying node clusters by matrix representation
(5) Drawing edges

4.4.1 Node Clustering

Cluster analysis or clustering is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. Clustering is a very important and useful method in graph drawing area[23][28], and also a common technique for statistical data analysis used in many other fields, such as machine learning, data mining, pattern recognition, image analysis, and bioinformatics.

Node clustering is a very important step in our research, and this kind of research has emerged in analyzing networks of many kinds, including the World Wide Web, citation networks, transportation networks, software call graphs, email networks, food webs, and social and biochemical networks[21][28][29]. Node clustering is always a computationally demanding job but it is an effective way to improve the readability of graphs especially large scale data.

The purpose of node clustering in our research is to make nodes with close relationship into groups of nodes within which connections are dense but between which they are sparser. For the same purpose, Newman developed a fast algorithm for detecting community structure with high quality [23], so in our search we decided to use his algorithm for node clustering (Figure 8).

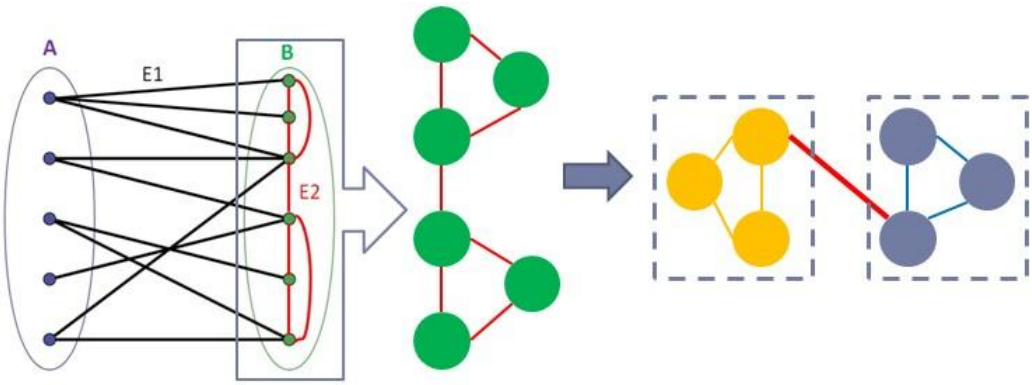


Figure 8 node clustering

4.4.2 Fixing nodes of set A

The same as related research “anchored map”, node of set A is fixed on a circumference as

anchors. It is obvious that anchors with great relationships should be arranged near each other, and it has been proved that the order of anchors has a great influence on the edge crossing and edge length when visualizing bipartite graph [19].

In this procedure, the main purpose is to make sure anchors with close relations are laid out as closely as possible to satisfy (R1) of 4.3, and in this part, the same as bipartite graph, we also found that the order of anchors has a great influence on the edge crossing and edge length which means (R4) and (R5) of 4.3 can be also satisfied by fixing anchors with close relations near to each other.

However, finding a good order in not an easy work. The most simple and straightforward way is to try all possible orders and find the best one, but it may be a very time–consuming job because of two reasons:

(1) The drawing result cannot be understood until the spring embedding model finishes, which is a very time costing procedure.
(2) The amount of all possible orders will be too large when the number of anchors increases.

So in order to find a good order, two parts are needed. First, an index which can indicate the goodness of the drawing result instead of running the spring embedding model. Second, an algorithm for finding an order with a good index is needed. In our research we developed an index which can indicate the goodness of drawing result for semi–bipartite graph. Details of this part will be discussed in chapter 4.

4.4.3 Layout

After the anchors are fixed (position will not be changed anymore), the position of elements in node set B (both single nodes and node clusters) need to be decided. As said before, elements of node set B are arranged at suitable positions by spring embedding [6].

And by the nature of the spring embedding, the free node will “move” to an appropriate position that expresses its relation to the connected anchors and free nodes. In this way, (R2) and (R3) of 4.3 can be satisfied.

4.4.4 Matrix Representation

This part is to visualize node clusters in matrix style, and it has been proved that visualizing dense graphs in a matrix representation is better than a node–link presentation. And since the matrices we are dealing with are all 0/1 matrices (edge exist or not), so we present nodes in line–column (Figure 1), and use color to display edges.

In our research we are focusing on the visualization of adjacent matrices. First we defined the aesthetic criteria for adjacent matrix representation which is the (R6) of 4.3, then, we developed an algorithm based on barycenter heuristic which will be discussed on chapter 6, last we will evaluate the proposed algorithm.

4.4.5 Drawing Edges

The last part of the proposed drawing style is to drawing edges. Edges within matrices are displayed by matrix representation so only edges between matrices or outside of matrices are drawn by straight lines.

Because of the character of the matrix, each node in the matrices will have four connecting points (Figure 9). We choose straight lines to represent the edges. And since each node within the matrices will have 4 connecting points, we will simply choose the nearest point and draw the edge. In this way, the overlapping between edge and matrices can be avoided and the length of edges can be reduced.

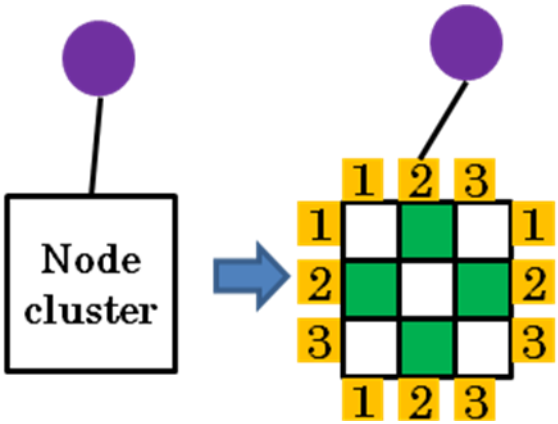


Figure 9 matrices with four connecting points

Chapter 5

Anchored Map for Semi−Bipartite Graph

In this chapter, we explain how to drawing a semi−bipartite graph in the anchored map style. First we will describe the aesthetic criteria, then how to find an anchor order satisfying the aesthetic criteria will be introduced.

5.1 Aesthetic Criteria

We employ the following aesthetic criteria for drawing semi—bipartite graph in anchored map style, and these aesthetic criteria are the 5 aesthetic criteria in 4.3.
(R1) Anchors with close relations are laid out as closely as possible.
(R2) Free nodes connected to common anchors are laid out as closely as possible.
(R3) Free nodes connected to each other are laid out as closely as possible.
(R4) Minimize the total length of edges
(R5) Minimize the number of edges−crossings.

5.2 Drawing Procedure

Anchored map of semi−bipartite graph will be laid out in two steps:
(Step1) Decide the order of anchors and fix anchors on the circumference at equal intervals.
(Step2) Arrange free nodes at suitable positions in relation to adjacent anchors and other free nodes.

The size of circumference (i.e., radius) will be decided in step 1. This size does not influence the quality of the layout but only the size of drawing result. The most important of step1 is to decide the order of anchors. Because after the order is decided, the position of both anchors and free nodes will be decided. The order of anchors has a great influence on the quality of the layout which can be seen in Figure 10. It is obvious that the edge−crossing and edge length will change a lot by different orders of anchor, Figure 10 are using the same data and the only difference is the anchor order (only a3 and a4 switched in this case). So we insist that the key to drawing an anchored map is to decide the order of anchors. How to decide the order of the anchors will be described in next section. In step 2, the position of free nodes will be decided by spring embedding with the restriction that the anchors will be fixed, so only the position of free nodes will change to a suitable place by spring embedding caused by both E_1 and E_2 edges. The spring embedding as well as the initial positions of the free nodes will have some

influence. However, those influences are negligible compared with the order of anchors.

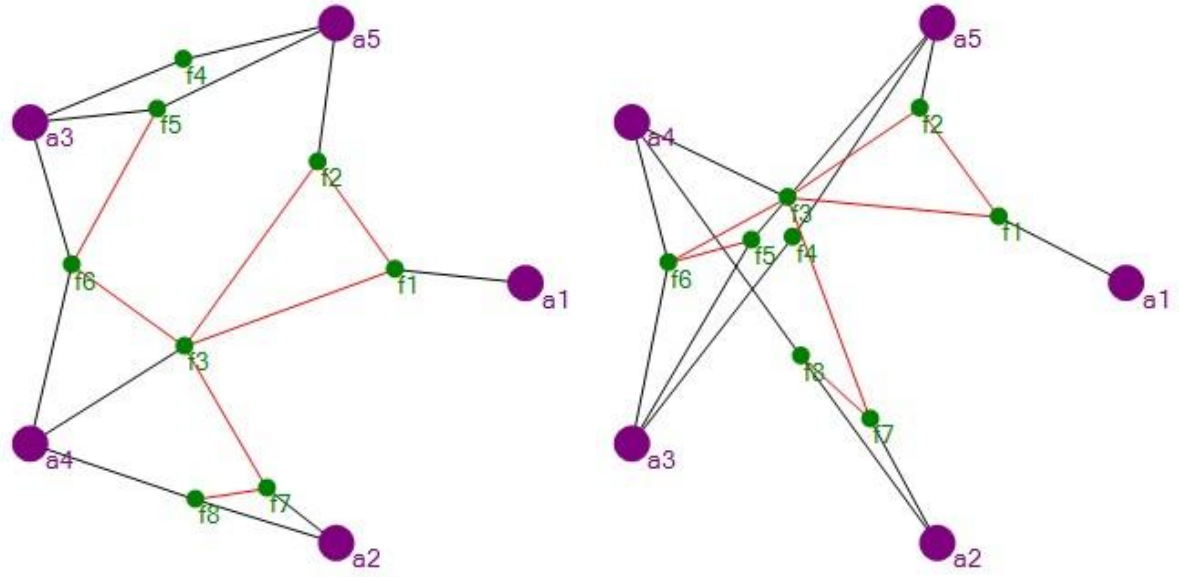


Figure 10 same data with different anchor order

5.3 Deciding the Anchor Order

In this section, we will discuss about how to decide the order of anchors. Here we want to emphasize that if anchors have its natural order, for example, when anchors are nodes presenting the days of week, we should arrange them as "Sunday", "Monday"... "Saturday". And if they don't have a natural order (actually in most cases), we need to find a good order for them.

The goodness of a certain anchor order can be evaluated only after the spring embedding has been processed. The simplest and most straightforward idea is to try all possible orders to find the optimal one. But this would require too much computing time. Suppose there are N anchors, the amount of all possible orders will be (N−1)! /2 (not N! because anchors are fix on a circumference and both clockwise and anticlockwise will return the same result), which means the computer has to run spring embedding (N−1)! /2 times to find the optimal order.

Two things are needed for deciding the order of anchors: first, an index for indicating the goodness of certain anchor order instead of performing spring embedding. Second an algorithm to search for an anchor order with good index is needed, instead of trying all possible (N−1)! /2 orders.

5.3.1 How to Define the Index

Misue has discussed several definitions of indexes for bipartite graph [13], and "the distance along the circumference of anchors" has been proved to be a reliable one. An index indicating "the closeness of anchors connected to common free nodes" has been

proved to be a good index for bipartite graph.

Semi–bipartite graphs are different from bipartite graphs because of the existence of E_2 edges, so the index of bipartite graphs cannot be used. An example is shown in Figure 4. Both of them have put anchors connected to common free nodes as closely as possible. However the drawing result is different because of the existence of E_2 edge, and Figure 4 right is obviously better then left.

In short, in order to decide the order of anchors for semi－bipartite graph, two requirements are needed. First, anchors connected to common free nodes are laid out as closely as possible. Second anchors which can be connected through by E_2 edges are laid out as closely as possible.

By extending the method of anchored map [12], we have defined an index for the anchor order of a semi–bipartite graph named “penalty”, and now we will explain the definition of penalty.

5.3.2 Penalty of Semi–Bipartite Graph

The penalty is a index of the goodness of drawing result, and the requirements for anchor order is to put related anchors as close as possible, and penalty is showing how well (or bad) this requirement is satisfied. In our research, we proposed two different definitions of “penalty” for semi–bipartite graphs which will be discussed later in this paper. We also evaluate these two definitions of penalty at the end of this chapter.

5.3.3 Search for the Optimal Penalty

After definition of penalty, we need to find a good order for penalty. Misue [19] has developed an algorithm for finding good anchor order on the circumference for bipartite graphs, and it has been proved to be a reliable one. In our research, we decided to use his algorithm for searching a sub–optimal penalty.

5.4 Definition of Penalty

As discussed before, we proposed two different definitions of “penalty” for a semi–bipartite graph. The goal for penalty is to indicate how far the related anchors are away from each other. So two things are needed for penalty:

(1) How far are two anchors away from each other
This can be easily understood since anchors are fixed on the circumference at equal intervals, so the distance between two anchors can be easily understood by the anchor order.

(2) How much are the two anchors related to each other
This part is not relatively hard to definite since anchors can be connected though by E_1 edges or both E_1 and E_2 edges (Figure 11). And for two anchors, there are may be

more than one path, for example, there are two paths between anchor a1 and a2 in Figure 11.

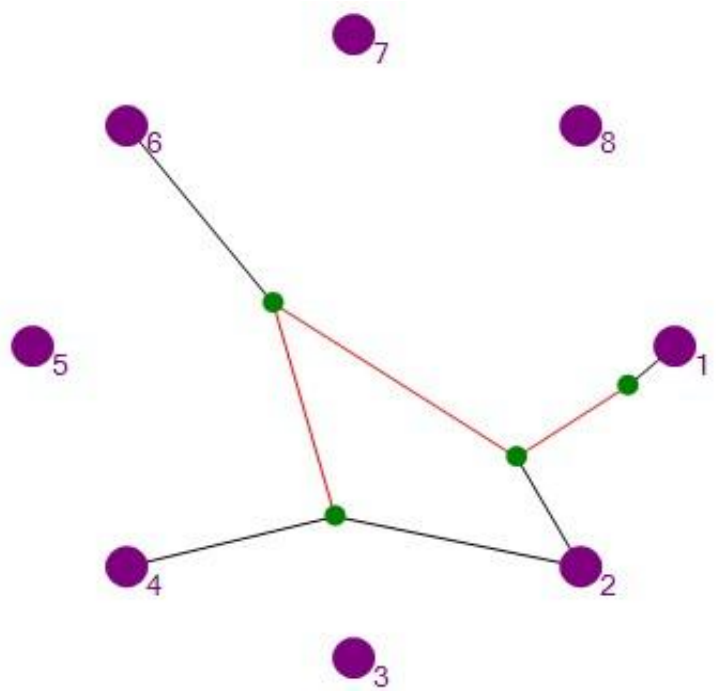


Figure 11 drawing example of semi–bipartite graph

To define the distance between anchors, first we will give some definition, and then we will discuss the two different definitions of "penalty".

Suppose that M is the number of anchors, that is, M = | A | .The anchors are arranged on the vertices of a regular N–gon (a polygon with N vertices). The vertices of the N–gon are labeled clockwise from 1 to M. It doesn't matter which vertex is chosen to be 1.

p(a) is the position of anchor a. and p : A → {1,2,3…N}

lm (i,j) is defined as the distance between anchor i, j

x= (p(i) − p(j) + N)mod N

y= (p(j) − p(i) + N)mod N

l_m (i,j) = min{x,y}

5.3.2 Shortest path

Since there may be more than one path between two anchors, so how to deal with multiple path becomes a problem, and in this definition we choose to definite how much two anchors are related to each other by the shortest path (least E_2 edge, since the number of E_1 edges are always 2).

The penalty of shortest path is defined as follows:

$$P = \sum_{i,j \in A, i!=j} g(i,j)$$

sp(i,j) is defined as the shortest path between anchor i and j.

$a(i,j) = \{ e \in E1 \mid e \in sp(i,j) \}$,

$b(i,j) = \{ e \in E2 \mid e \in sp(i,j) \}$

$d(i,j) = w_1 * |a(i,j)| + w_2 * |b(i,j)|$

$g(i,j) = l_m(i,j) / d(i,j)$

5.3.3 All paths

In this definition, we choose to definite how much two anchors are related to each other by all possible paths. For one single path, we calculate the penalty the same as 5.4.1, the only difference is all possible path should be calculated by this method.

$$P = \sum_{i,j \in A\ i!=j,} t(i,j)$$

p(i,j) is defined as a possible path between anchor i and j.

$a(i,j) = \{ e \in E1 \mid e \in sp(i,j) \}$

$b(i,j) = \{ e \in E2 \mid e \in sp(i,j) \}$

$d(i,j) = w_1 * |a(i,j)| + w_2 * |b(i,j)|$

$q(i,j) = l_m(i,j) / d(i,j)$

$$t(i,j) = \sum q(i,j)$$

5.3.4 Definition of w_1 and w_2

As described before, w_1 and w_2 is the weight of E_1 edge and E_2 edge. It is obvious that E_1 edge should have more influence than E_2 edge which has been proved. And in our research, we choose w_1 to be 1, and w_2 to be 2. In this way, E_1 edge will have more influence than E_2 edge, and has been proved in most cases, it will bring better results than treating those two kinds of edges the same way.

5.5 Evaluation

The main contribution of our extended anchored map is the definition of “penalty”. In order to find out how well the proposed two penalties work. First, we want understand how “penalty” works in different anchor size, because it is obvious that the size of anchors may have some influence. Then, how penalty can indicate the two most important aesthetic criteria — edge length and edge crossing, will be showed by calculating the correlation between them. Last, we will compare the result of the two proposed definitions of “penalty”.

5.5.1 Design of Experiment

First we made 100 random graphs for each anchor number of 10, 15, for each of these random graphs, we recorded results of 1000 different anchor orders (not all possible orders because there are (N−1)! /2 orders when anchor number is N), and calculate the correlation between penalty and edge−crossing and edge−length for each graph, and see how well the penalty can indicate the goodness of drawing result. And it is obvious that

the anchor order for both the least edge–crossing and the shortest edge–lengths may not exist.

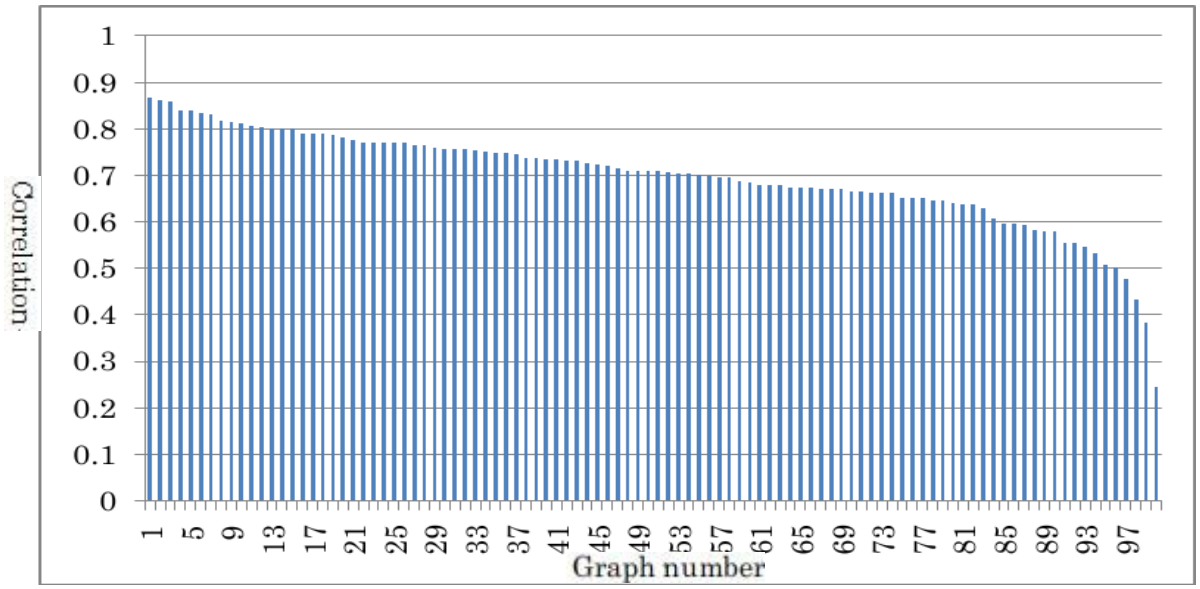
5.5.2 How to Make Experiment Data

Let n and p be parameters to generate a random graph, n denotes the number of anchors, and p denotes the appearance probability of free nodes. We try 2^n times, and for each time, the appearance probability of a free node is p, so in this way, the number of free nodes will be decided. After free nodes are decided, each free node will have a certain possibility to have edges between anchors (pe1) and other free nodes (pe2).When n = 10, we change p from 0.03 to 0.05, and set pe1 = 0.1, pe2 = 0.02. When n=15, we change p from 0.0016 to 0.002, and set pe1 = 0.06, pe2 = 0.02. In this way, random graphs will not be too dense or too sparse. After that, free node with degree < 2 will be deleted, since they will have no influence on the order of anchors.

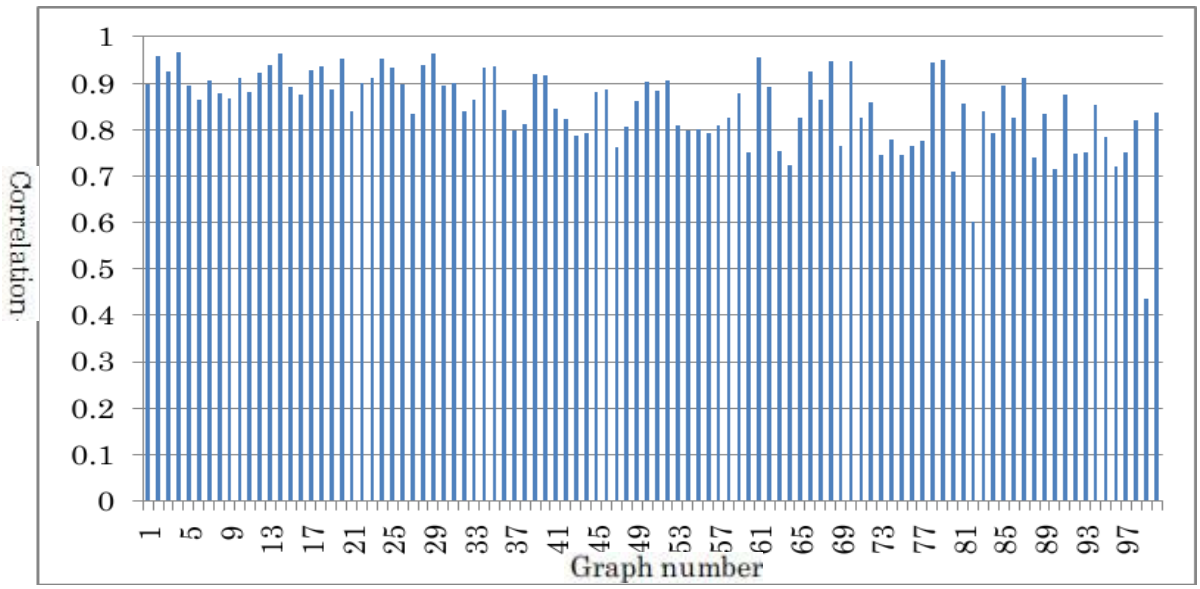
5.5.3 Result of “Shortest path”

First we will show the correlation result of graphs with 10 anchors, then the result of graphs with 15 anchors. We will show the correlation result both between penalty edge–crossing and penalty edge–length. In all figures the vertical axis is the coefficient of correlation and the horizontal axis is the graph number.

Figure 12 (a) shows the result of correlation between penalty and edge–crossing in descending order and it can be easily seen that correlation are over 0.6 in more than 80 graphs. Realized that, the correlation between penalty and edge–crossing is independent from correlation between penalty and edge–length, in order to see how penalty work, Figure 12 (b) shows the result of the correlation between penalty and edge–length with the same vertical axis order. And it is can be seen that in no.99 graph, both of the correlation are only around 0.4 which is not good enough, and for no.100 graph, even though, the correlation between penalty and edge－crossing is only 0.24 but the correlation between penalty and edge–length is about 0.84.



(a) "short path" correlation result between penalty, edge–crossing



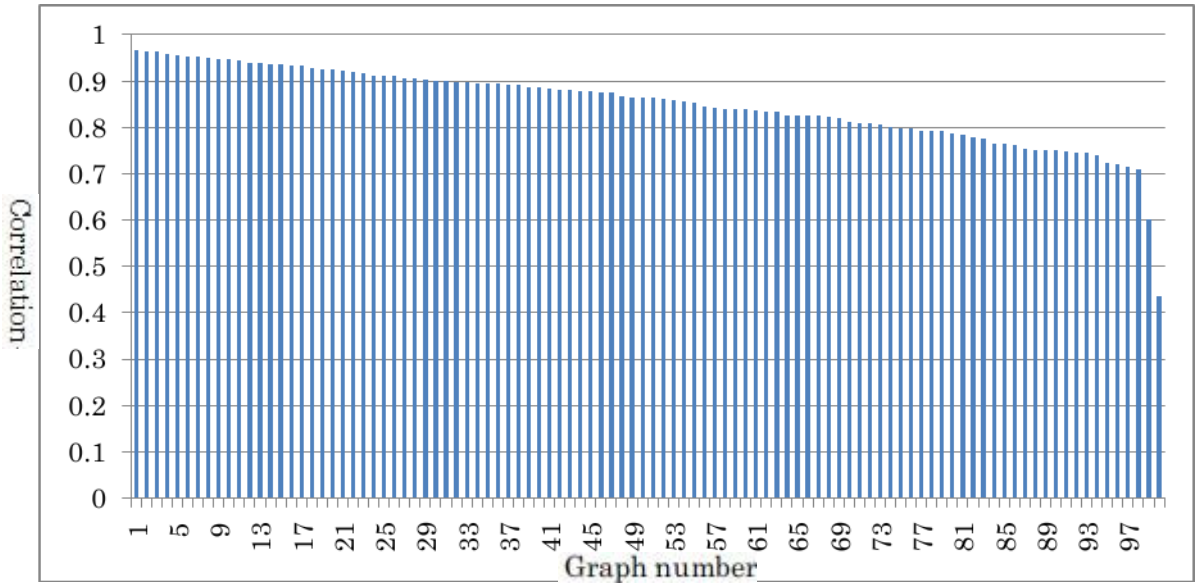
(b) "short path" correlation result between penalty, edge–length

Figure 12 "short path" correlation result in descending order (penalty, edge–crossing) of graphs with 10 anchors

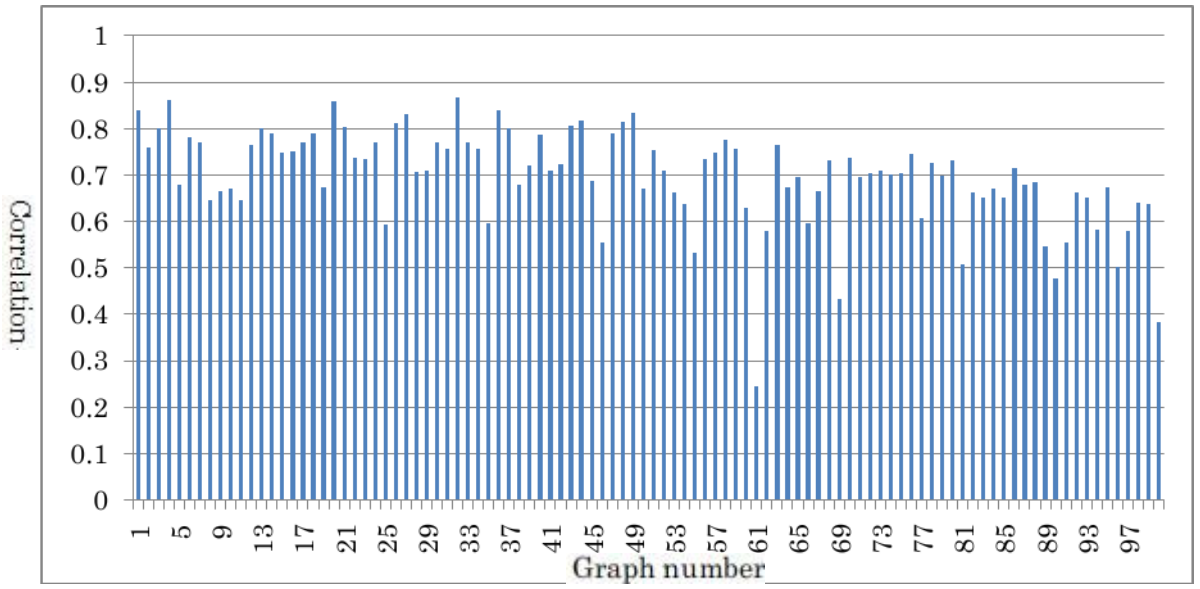
Figure 13 shows the same data of Figure 12 in different vertical axis order. Figure 13 (a) shows the correlation between penalty and edge–length in descending order and it is it can be easily seen that the result is better than edge–crossing's, correlation is over 0.8 in about 70 graphs. This is quite predictable because of the existence of two different kinds of edges.

Figure 13 (b) shows correlation between penalty and edge–crossing in same vertical axis order of (a). And two correlations seems quite independent from each other. In no. 66 graph, the correlation between penalty and edge–length is about 0.84 while the

correlation between penalty and edge–crossing is only 0.24.



(a) short path" correlation result between penalty, edge–length



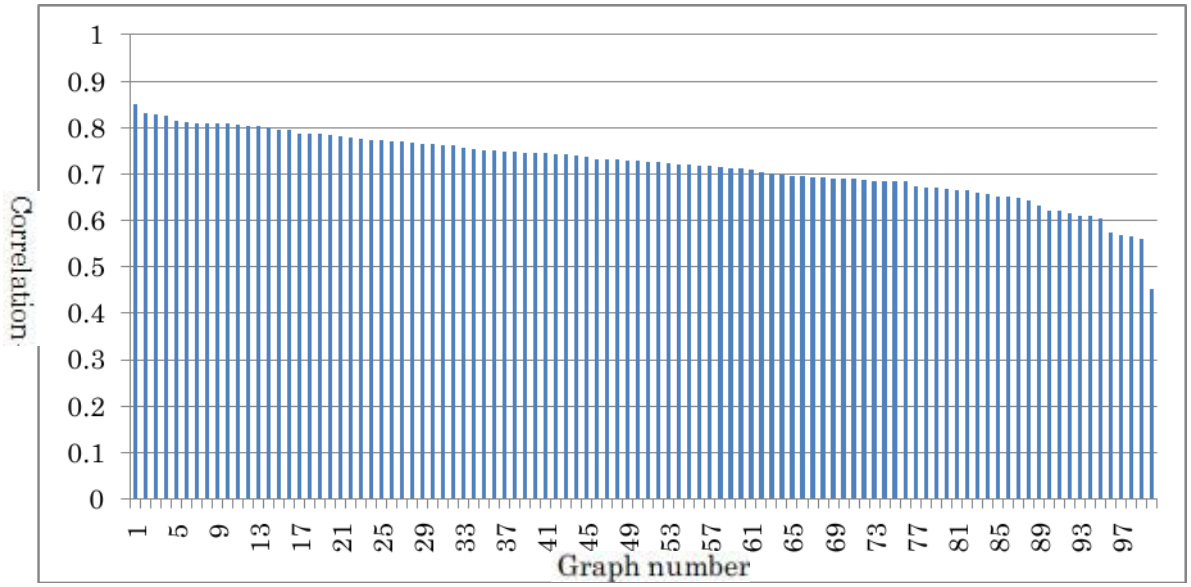
(b) short path" correlation result between penalty, edge–crossing

Figure 13 "short path" correlation result in descending order (penalty, edge–length) of graphs with 10 anchors

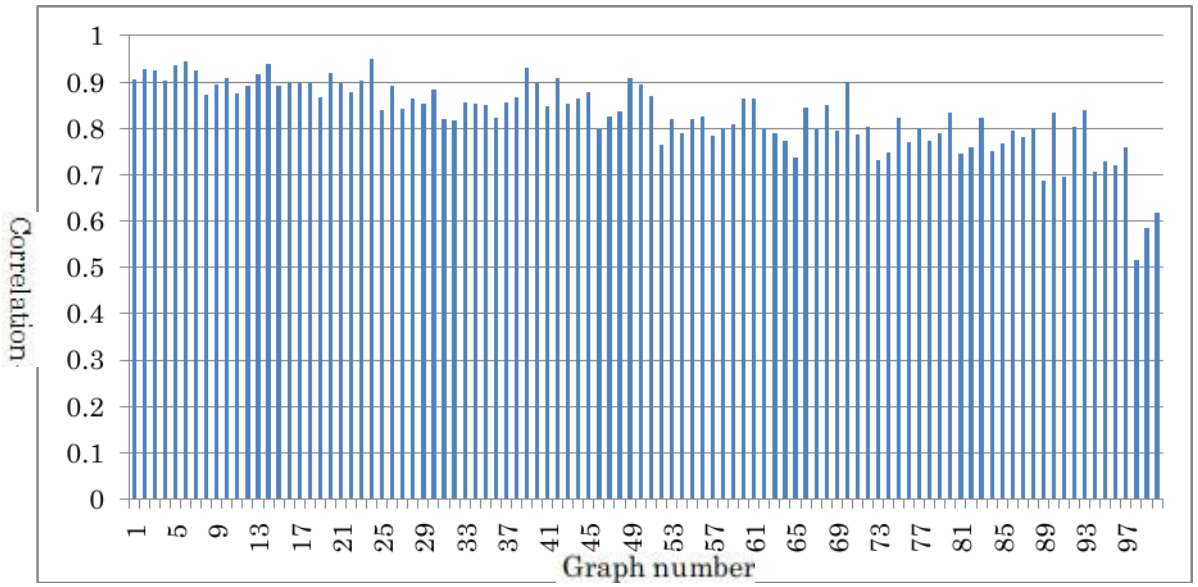
Then we will show the correlation result of graphs with 15 anchors, the same as the former result, Figure 14 (a) is the result of correlation between penalty and edge–crossing, Figure 14 (b) is the result of correlation between penalty and edge–length. The vertical axis shows the coefficient of correlation while the horizontal axis shows the graph number.

Figure 14 (a) shows the result of correlation between penalty and edge–crossing in descending order and it can be easily seen that the correlation are over 0.65 in more than

80 graphs which is a better result than result of graphs with 10 anchors. Figure 14 (b) shows the result of correlation between penalty and edge–length with the same vertical axis order.



(a) short path" correlation result between penalty, edge–crossing

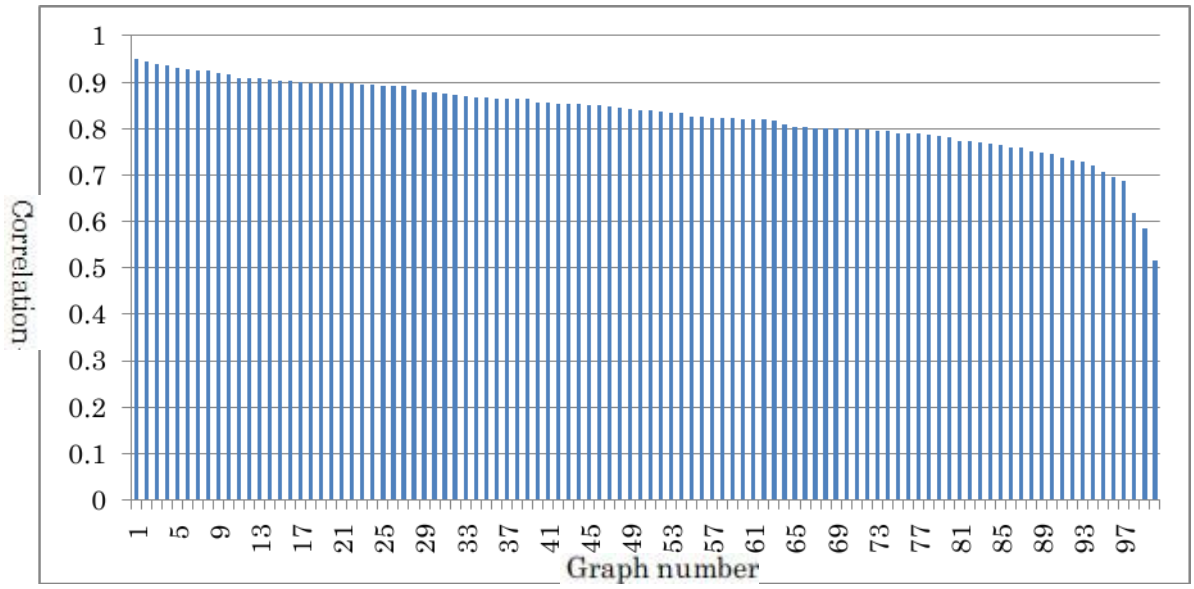


(b) short path" correlation result between penalty, edge–length

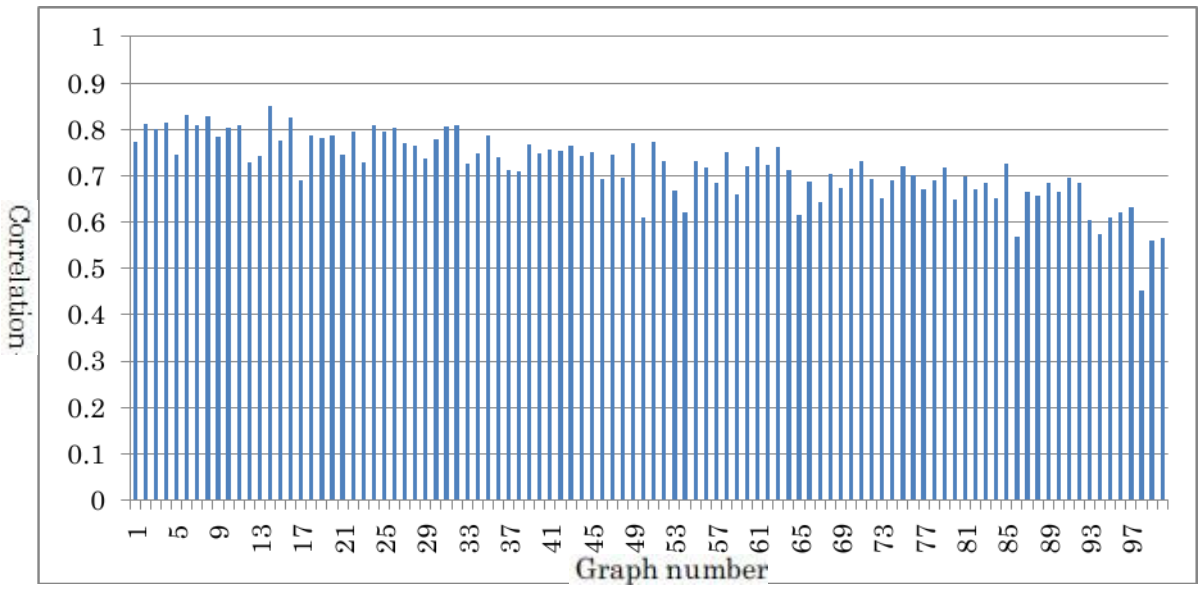
Figure 14 "short path" correlation result in descending order (penalty, edge–crossing) of graphs with 15 anchors

Figure 15 shows the same data of Figure 14 in different vertical axis order. Figure 15 (a) shows the correlation between penalty and edge–length in descending order, the correlation is over 0.8 in about 70 graphs. Figure 15 (b) shows correlation between penalty and edge–crossing in same vertical axis order of (a).

Generally, the result of graphs with 15 anchors is better than graphs with 10 anchors, and this will be discussed later in the conclusion part of this chapter.



(a) short path" correlation result between penalty, edge–length



(b) short path" correlation result between penalty, edge–crossing

Figure 15 "short path" correlation result in descending order (penalty, edge–length) of graphs with 15 anchors

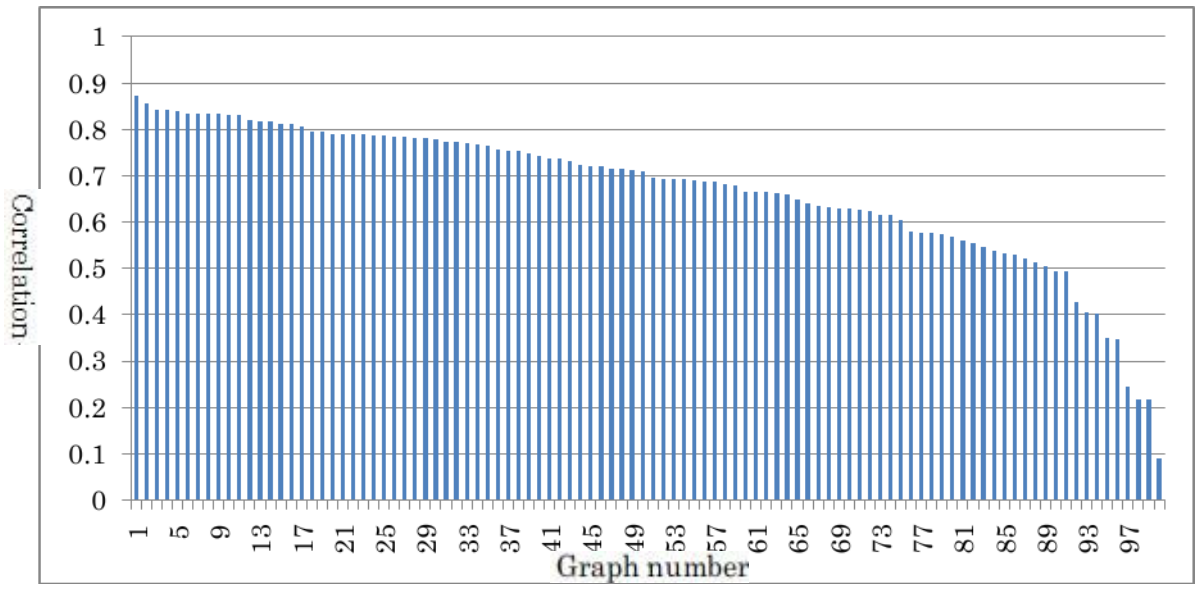
5.5.4 Result of "All paths"

The same as with "shortest path", first we will show the result of anchor 10. The vertical axis shows the coefficient of correlation while the horizontal axis shows the graph number (sorted from better result to worse result). Figure 16 (a) is the result of correlation between penalty and edge–crossing, Figure 16 (b) is the result of correlation between penalty and edge–length.

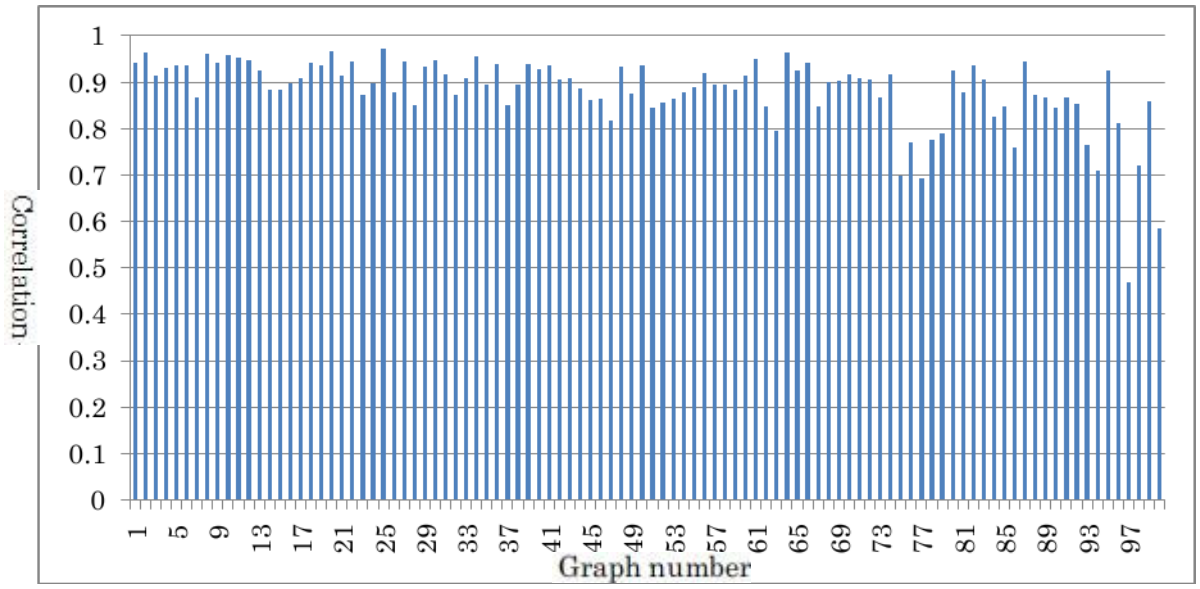
Figure 16 (a) shows the result of correlation between penalty and edge–crossing in descending order, although the correlation is over 0.6 in 75 graphs, this result is relatively worse than "shortest path" (Figure 12), since in over 10 graphs the correlation

is less than 0.4.

Figure 16 (b) shows the result of correlation between penalty and edge–length with the same vertical axis order. And it can be seen that no.97 graph's result is not so good, the correlation between penalty and edge–crossing is only 0.25, and correlation between penalty and edge–length is 0.47.



(a) "all paths" correlation result between penalty, edge–crossing

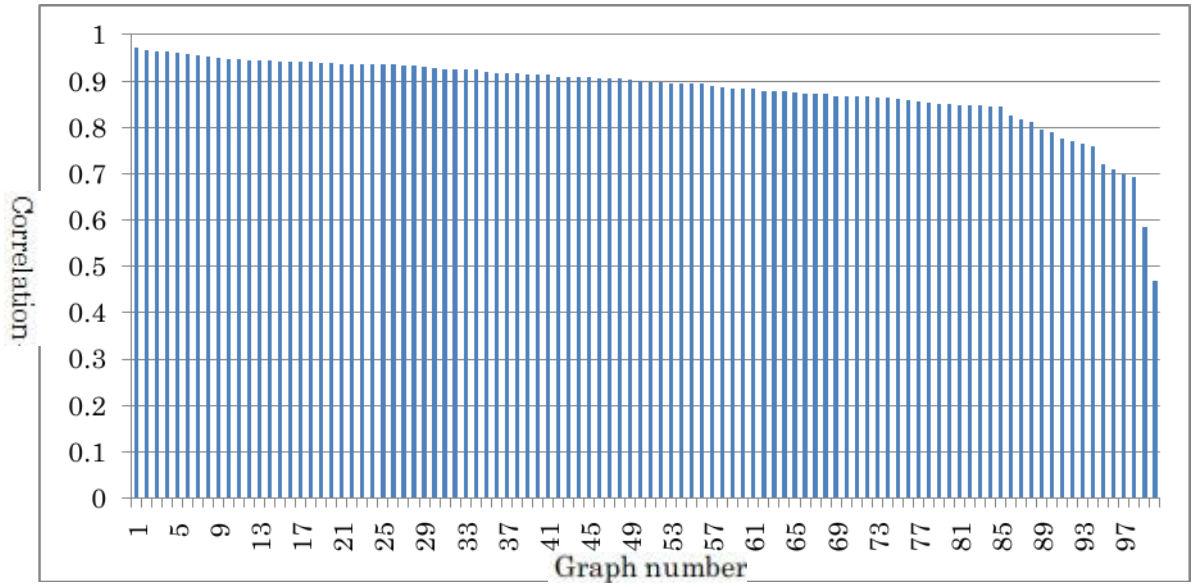


(b) "all paths" correlation result between penalty, edge–length

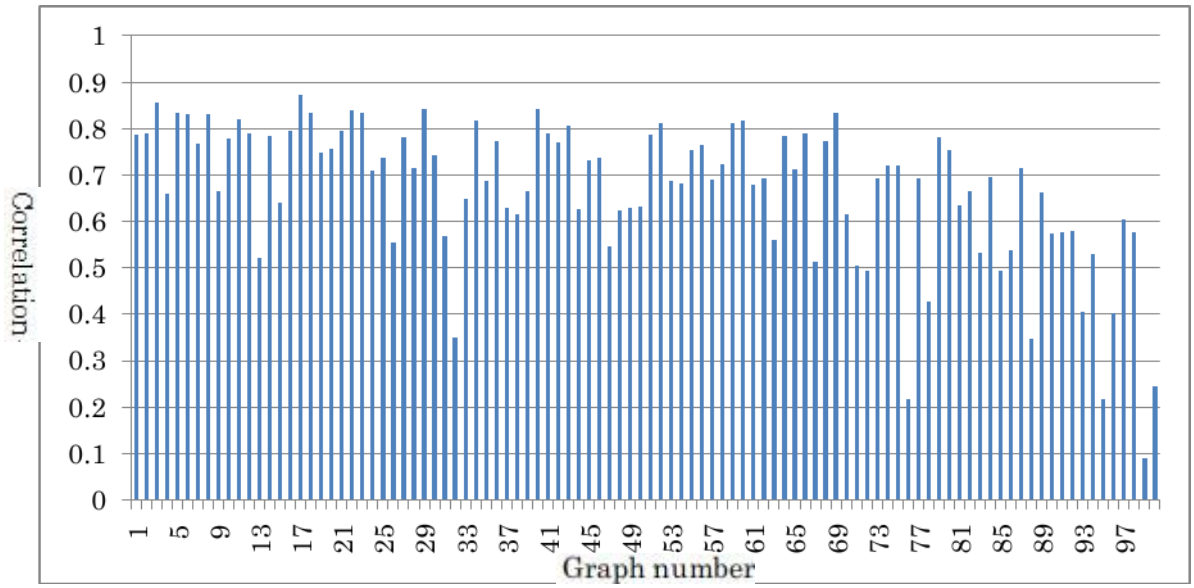
Figure 16 "all paths" correlation result in descending order (penalty, edge–corssing) of graphs with 10 anchors

Figure 17 shows the same result of Figure 16 in different vertical axis order. Figure 17 (a) shows the correlation between penalty and edge–length in descending order, and it can be understood that "all paths" works better than "shortest path" (Figure 13) since correlation is more than 0.8 in 87 graphs. Figure 17 (b) shows result of correlation

between penalty and edge–crossing with the same vertical axis order, and it is easy to see that these two relations are quite independent.



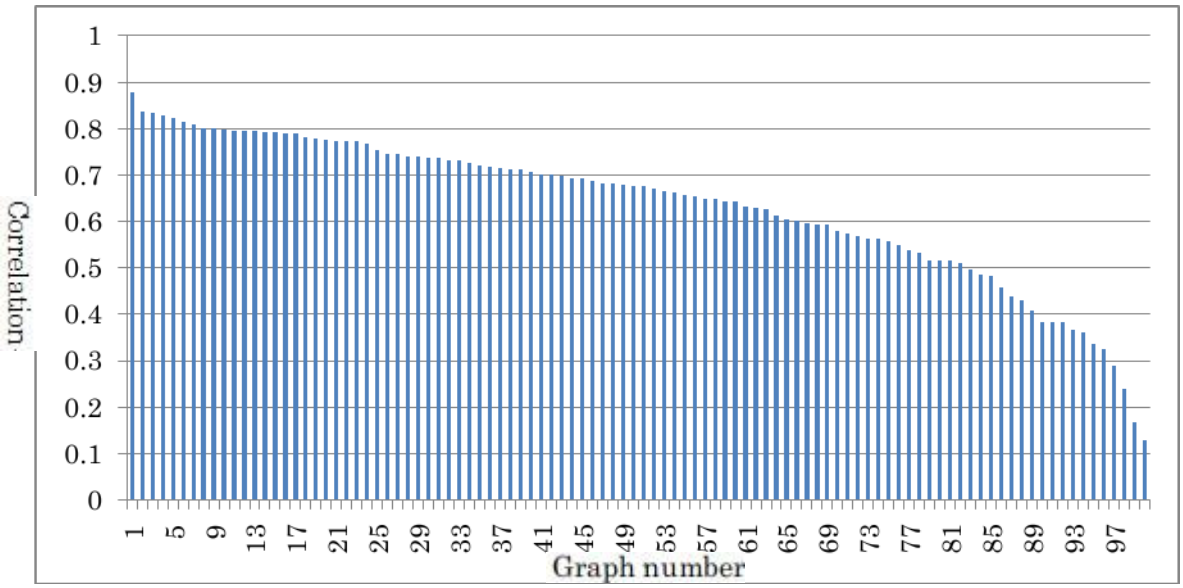
(a) "all paths" correlation result between penalty, edge–length



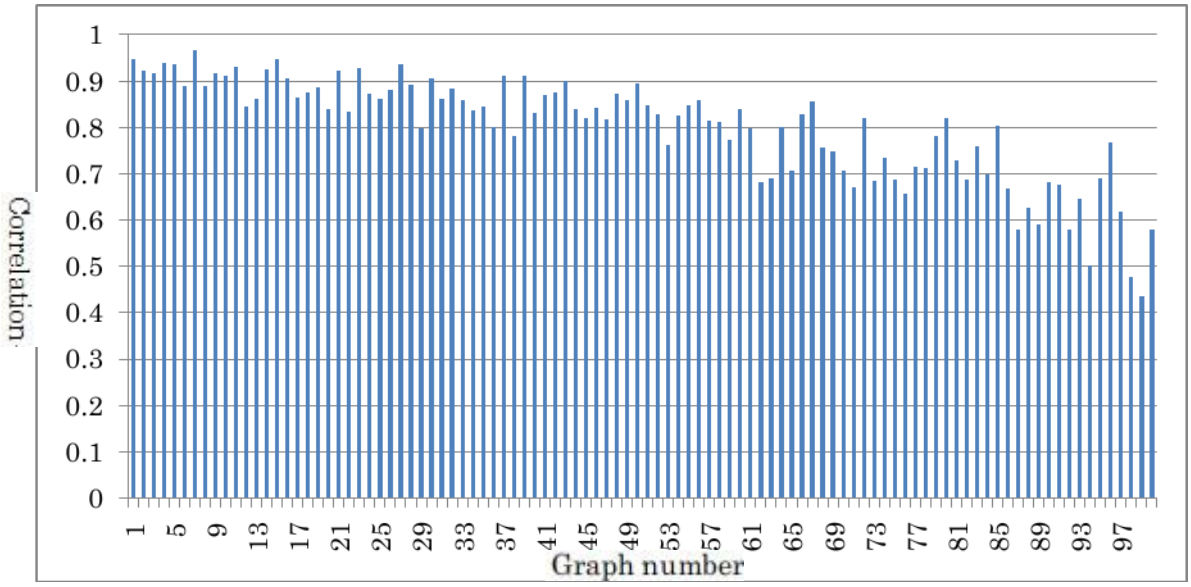
(b) "all paths" correlation result between penalty, edge–crossing

Figure 17 "all paths" correlation result in descending order (penalty, edge–length) of graphs with 10 anchors

Lastly we will show the result of graphs with 15 anchors. Figure 18 (a) shows the correlation between penalty and edge–crossing in descending order where Figure 18 (a) shows the correlation between penalty and edge–length in the same order. And the result is not as good since no more than 65 graphs have a correlation more than 0.6 in (a).



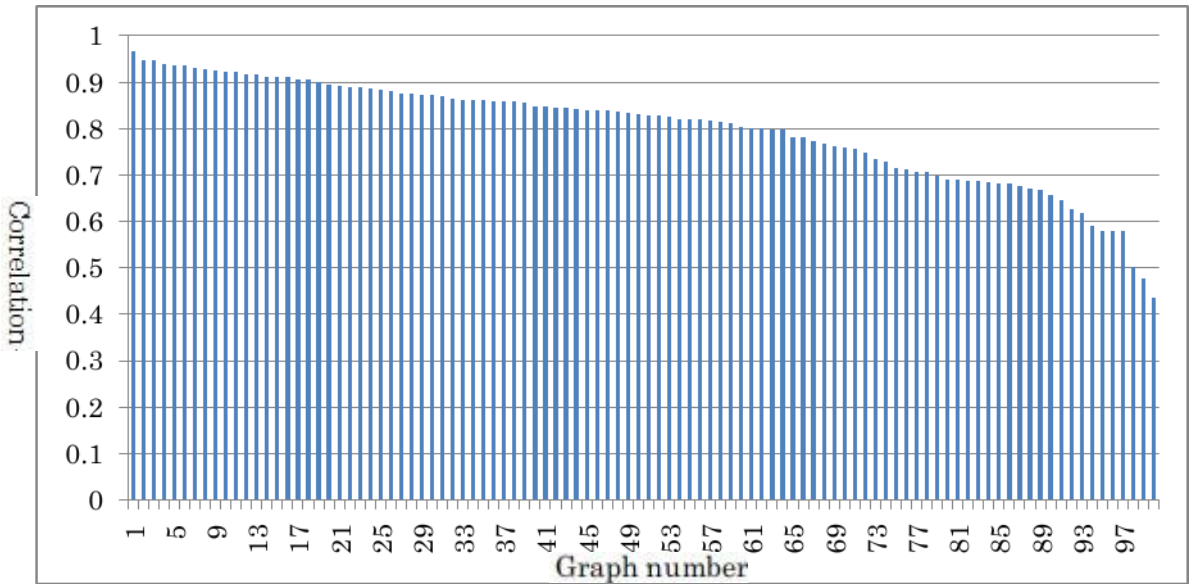
(a) "all paths" correlation result between penalty, edge–crossing



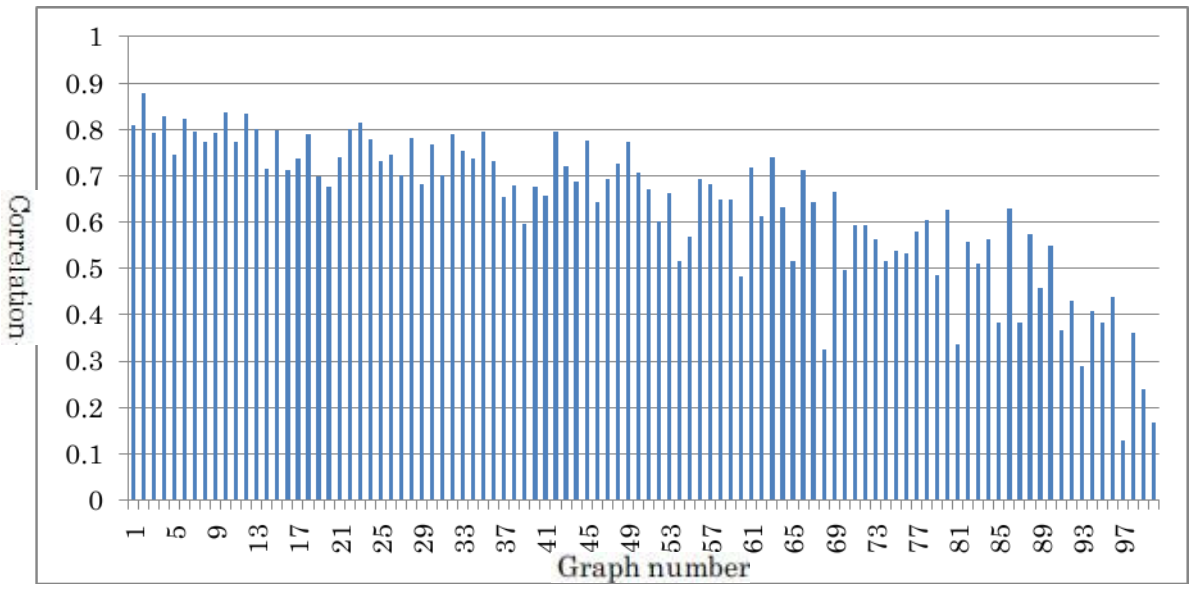
(b) "all paths" correlation result between penalty, edge–length

Figure 18 "all paths" correlation result in descending order (penalty, edge–crossing) of graphs with 15 anchors

Figure 19 show the same result of Figure 18 in different vertical axis order. It can be seen that in most of graphs can bring a good result except No.100 graph (No. 99 graph of Figure 18) where correlation between penalty edge–crossing is only 0.16 and correlation between penalty edge–length is 0.43.



(a) "all paths" correlation result between penalty, edge–length



(b) "all paths" correlation result between penalty, edge–crossing

Figure 19 "all paths" correlation result in descending order (penalty, edge–length) of graphs with 15 anchors

5.5.5 Conclusion

Generally, penalty is a good index for indicating both edge crossing and edge length in most cases, since in most cases, the correlations are over 0.6, and compared with edge crossing, penalty has better correlation with edge length, which is predictable since two kinds of edges exist and the edge crossing problem is relatively hard to estimate.

By analysis of the results of correlation we found that, in most cases, "shortest path" works better than "all path", especially in graphs with 15 anchors. And "all paths" method will be more costly since all possible paths should be calculated, so in our research, we prefer "shortest path" method.

5.6 Drawing Examples

In this part, we will show the effectiveness of extended anchored map by showing some drawing examples. Figure 20 shows a drawing example of a simple graph where only five anchors and five free nodes exist. It is obvious that the proposed method can find a very good order for this simple graph, after reordering, related anchors are fixed near to each other, edge crossing disappears, and edge length is shortened. Figure 21 and Figure 22 shows the drawing example of a relatively complex graph, Figure 21 shows graph in a random anchor the anchor order in Figure 22 is decided by the proposed method. It is obvious that the readability has been improved a lot after anchor reordering by the proposed method.

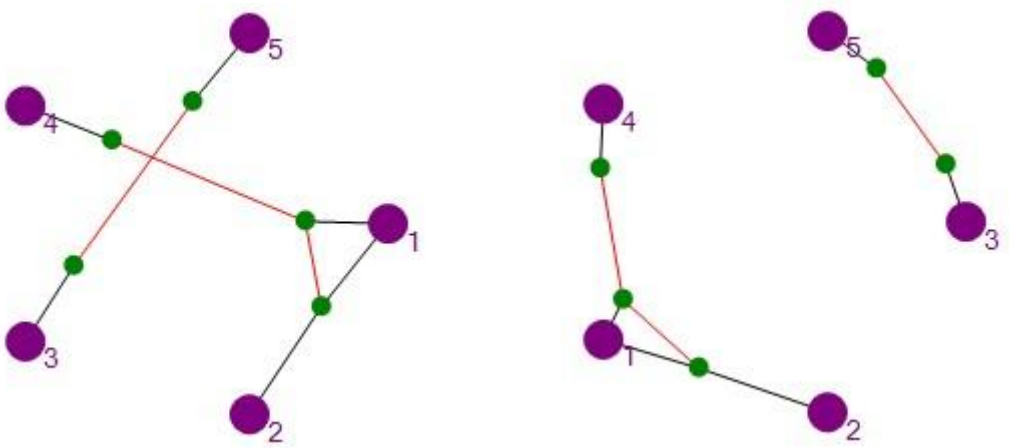


Figure 20 drawing example of a simple graph (left: random anchor order, right: anchor order decided by proposed method)

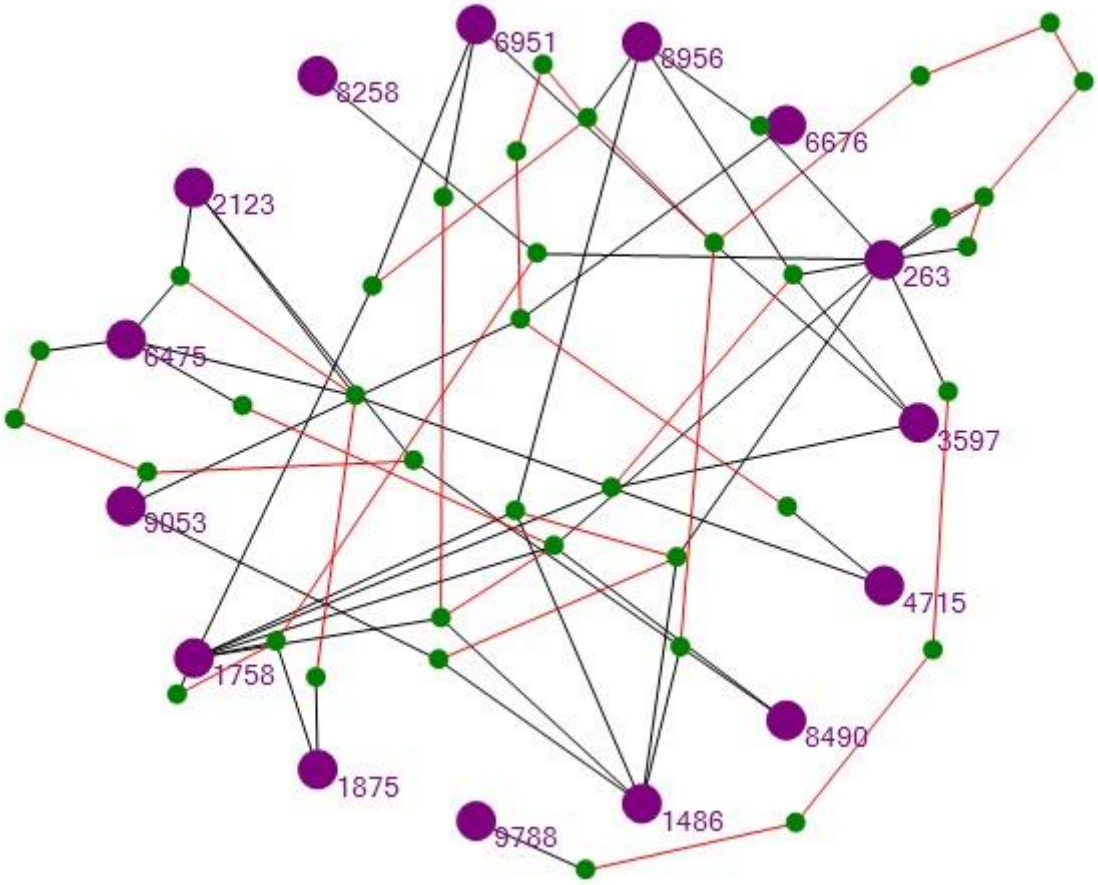


Figure 21 drawing example in random anchor order

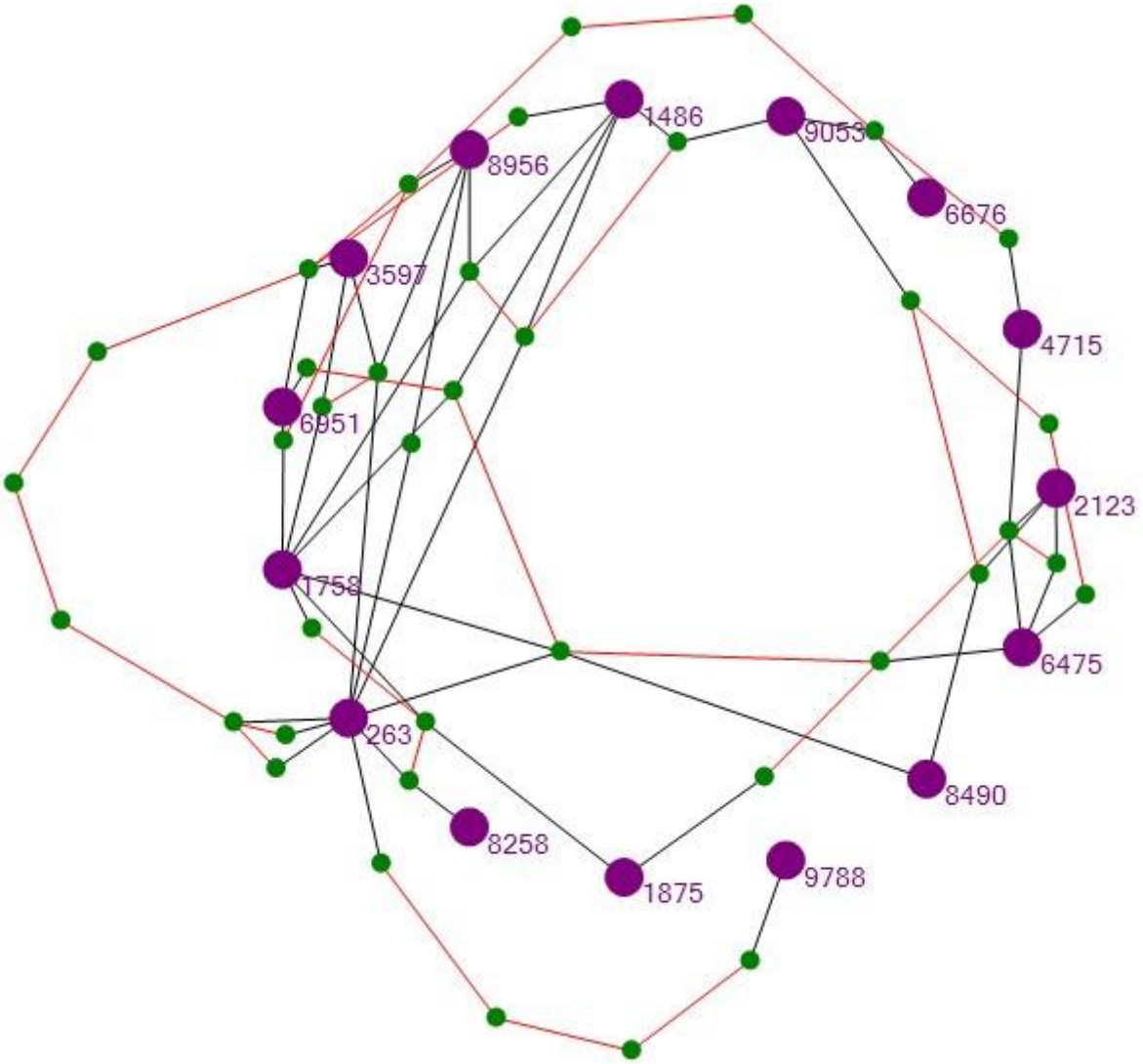


Figure 22 drawing example with anchor order decided by proposed method

Chapter 6

Matrix Representation

In this chapter, we explain how to draw node clusters by matrix representation. First we will discuss the aesthetic criteria of matrix representation of our research, then we will introduce an adjacent matrix reordering algorithm based on barycenter heuristic, last ,we will show the effectiveness of proposed algorithm.

6.1 Aesthetic Criteria of Matrix Representation

We employ the following aesthetic criteria for matrix representation which is the same aesthetic criteria of 4.3:
(R6) Free nodes (within single matrix) with close relations are placed near each other as close as possible.

Existed matrix ordering algorithms try to optimize an objective function useful for some network related operation such as Bandwidth, Minimum Linear Arrangement (MinLA), Cutwidth, Modified Cut, Vertex Separation, Sum Cut, Profile, Edge Bisection and Vertex Bisection[13]. These algorithms find a linear order of the vertices of a graph that optimizes either a function of the edge length (the distance between the two nodes), or of the number of crossings of the edges. Exact solutions to these functions are all NP—complete but some have good polynomial time approximations. Among these functions, some have been used for matrix visualization. Reducing the bandwidth is related to diagonalizing the matrix, a goal expressed by Bertin. And it consists in finding an order that minimizes the maximum edge length.

In our research, the matrix ordering problem is quite different from other related researches. First, the drawing object is diagonal matrices. Second the matrices are groups of nodes with close relations.

Since the main purpose for matrix representation is to revealing the connecting pattern between nodes within matrices, we considered that, if nodes with close relations are put each other as close to each other as possible, the readability of matrices may be better. To test whether a particular matrix order is good we define a quality function Q as follows:

$$Q = \sum_{f \in M} \sum_{f1,f2 \in c(f)} |p(f1) - p(f2)|$$

M represents the node set (matrix) and f represents the node inside.

$M = \{f_1, f_2 \ldots f_n\}$, $|M| = n$

$p(f)$ is the position of node f in matrix, where $p: M \rightarrow \{1,2,3 \ldots n\}$

$C(f)$ is defined as the node set inside matrix which have connection with node f.

$C(f) = \{v \in M \mid (v,f) \in E2 \wedge (f,v) \in E2\}$ or $C(f) = \{v \in M \mid \{v,f\} \in E2\}$

So the purpose of matrix ordering in part turns out to be a problem of finding an order with small Q. The most simple and straightforward way is to choose the order with smallest Q from all possible orders, which is impossible when the node size of matrix becomes large. In our research we developed a reordering algorithm based on the well known barycenter heuristic.

6.2 Barycenter Heuristic Based Algorithm

Barycenter heuristic is first proposed by Sugiyama et al. at 1981 for drawing hierarchical structures [30]. Makinen and Siirtola introduce the barycenter heuristic (Figure 23) as an efficient tool for manipulating the reorderable matrix by considering the ordering of the matrix as a bipartite graph drawing problem [22].

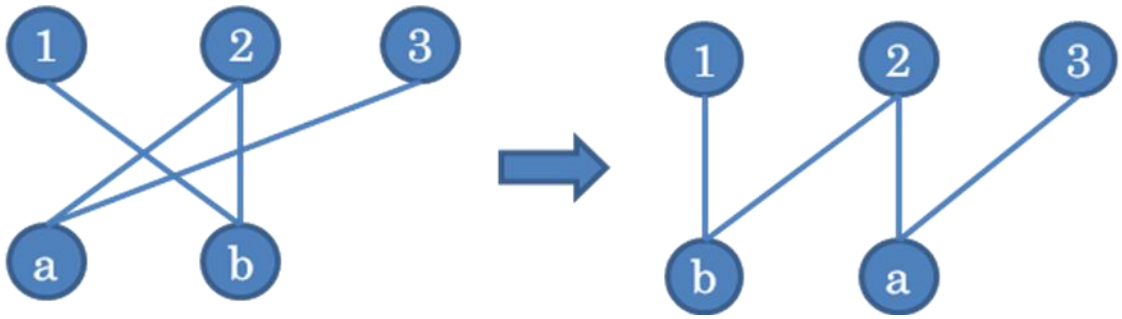


Figure 23 barycenter heuristic example

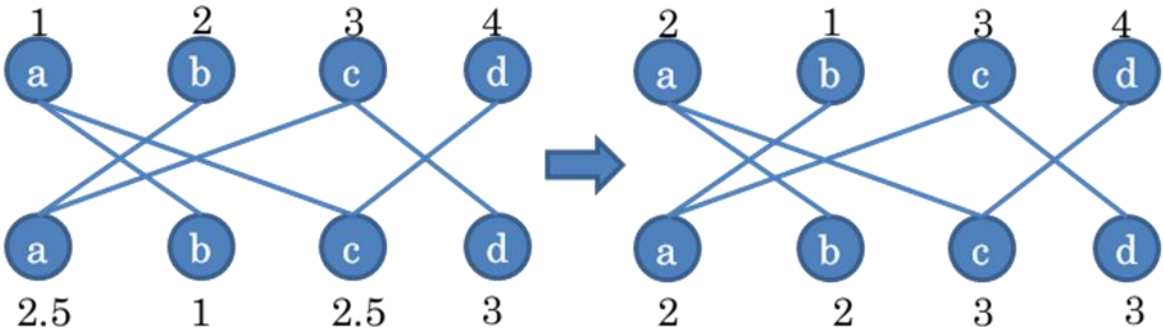


Figure 24 an ordering example of proposed algorithm

In the barycenter heuristic, the nodes will be ordered according to the averages of their adjacent nodes in the opposite node set. Repeating this ordering process in turns in the two node sets, may reach to an ordering of nodes that minimize the number of edge

crossings.

Barycenter heuristic seems to be a very good way to put node near each other, but it cannot be used directly since the drawing object of our research are diagonal matrices which cannot be treated as bipartite graphs. So we developed an algorithm based on barycenter heuristic to reorder diagonal matrices.

The same as brycenter heuristic, the ordering of nodes depends on the adjacent nodes, but the difference is that only one node set exists (Figure 24). In Figure 24, we can see that 4 nodes "a, b, c, d" exist with a default ordering "1, 2, 3, 4" shown above, node "a" is connected with "b, c", so the averages of adjacent nodes is 2.5. In the same way, we can see that the averages of adjacent nodes for node b, c, and d are 1, 2.5, and 3. The order of nodes "a, b, c, d" will be decided by the averages of adjacent nodes "2.5, 1, 2.5, 3", and after ordering is changed, the ordering of "a, b, c, d" as well their averages of adjacent nodes will also change.

By experiment, we found that repeating this ordering process may not bring better results, it may get worse sometimes or run into a loop. So in our research, we try repeating the ordering process n times (n is number of nodes inside matrix) and choose the order with smallest Q. In this way, we hopefully find a good ordering. The algorithm is given as follow:

Algorithm:
Given a matrix with n nodes as well as their connecting relations
Repeat n times
    For each node compute the average of its adjacent nodes
    Order the nodes by the averages of adjacent nodes
    Calculate and record the Q.
Find the ordering with best Q in n results.
End

To improve the effectiveness of proposed algorithm, we made an evaluation experiment which will be discussed later.

6.3 Evaluation

6.3.1 Design of Experiment

The purpose of this experiment is to evaluate the proposed matrix ordering algorithm, to see how much the reordering result can satisfy the aesthetic criteria described in 6.1.

For each node number of 5, 6, and 7, we made 1000 random matrices and see how well is the ordering decided by proposed algorithm. The vertical axis shows the top % of proposed result in all possible ordering, 0 means optimal result and 100 means the worst

result. The horizontal axis shows the data number (sorted from worse result to better result)

6.3.2 Result of Experiment

First, we will show the result of matrices with 5 nodes in Figure 25. It shows that in more than 70% random matrices, our proposed algorithm found the optimal result. Even though in some rare case which our algorithm does not work so well, the result is in the top 45%.

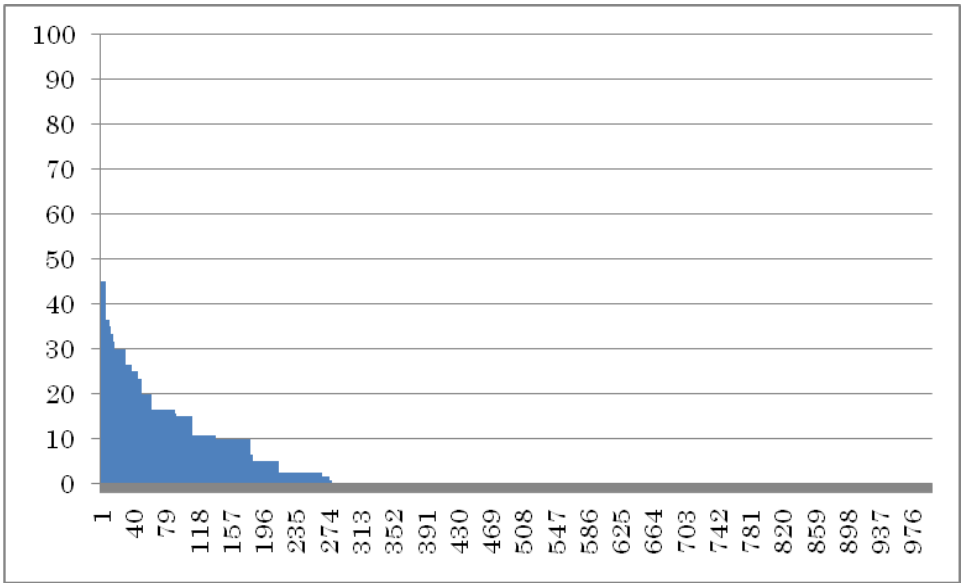


Figure 25 result of matrices with 5 nodes

Figure 26 shows the result of matrices with 6 nodes, it is also shows a very good result where in almost 60% random data, the proposed algorithm can reach to an optimal result. The worst result is still in top 45%.

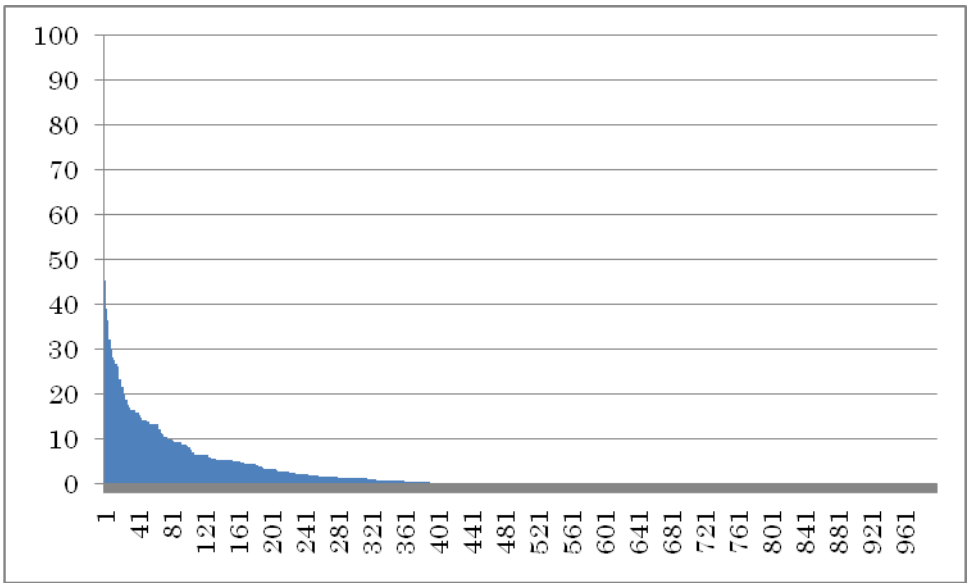


Figure 26 result of matrices with 6 nodes

Figure 27 shows the result of matrices with 7 nodes where in about 65% of random data, proposed algorithm can reach to an optimal result.

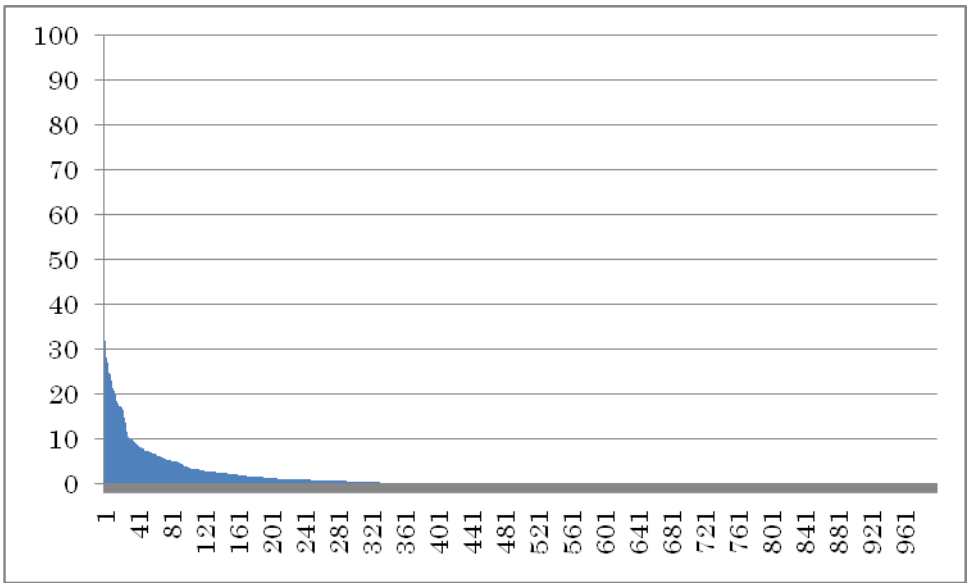


Figure 27 result of matrices with 6 nodes

6.4 Conclusion

From the experiment result, we can conclude that, proposed algorithm can satisfy the aesthetic criteria described in 6.1 well. Even if it cannot find the optimal result in some rare cases, the result is still not bad.

Here we will show a single result of matrix reordering, Figure 28 shows the graph with random matrix order, and Figure 29 shows the drawing result of same graph while using proposed matrix reordering algorithm. It is obvious that after putting related nodes close together, the readability of matrix is thought to be improved. It is considered that, the readability of matrices after reordering is thought to be better than random matrix ordering.

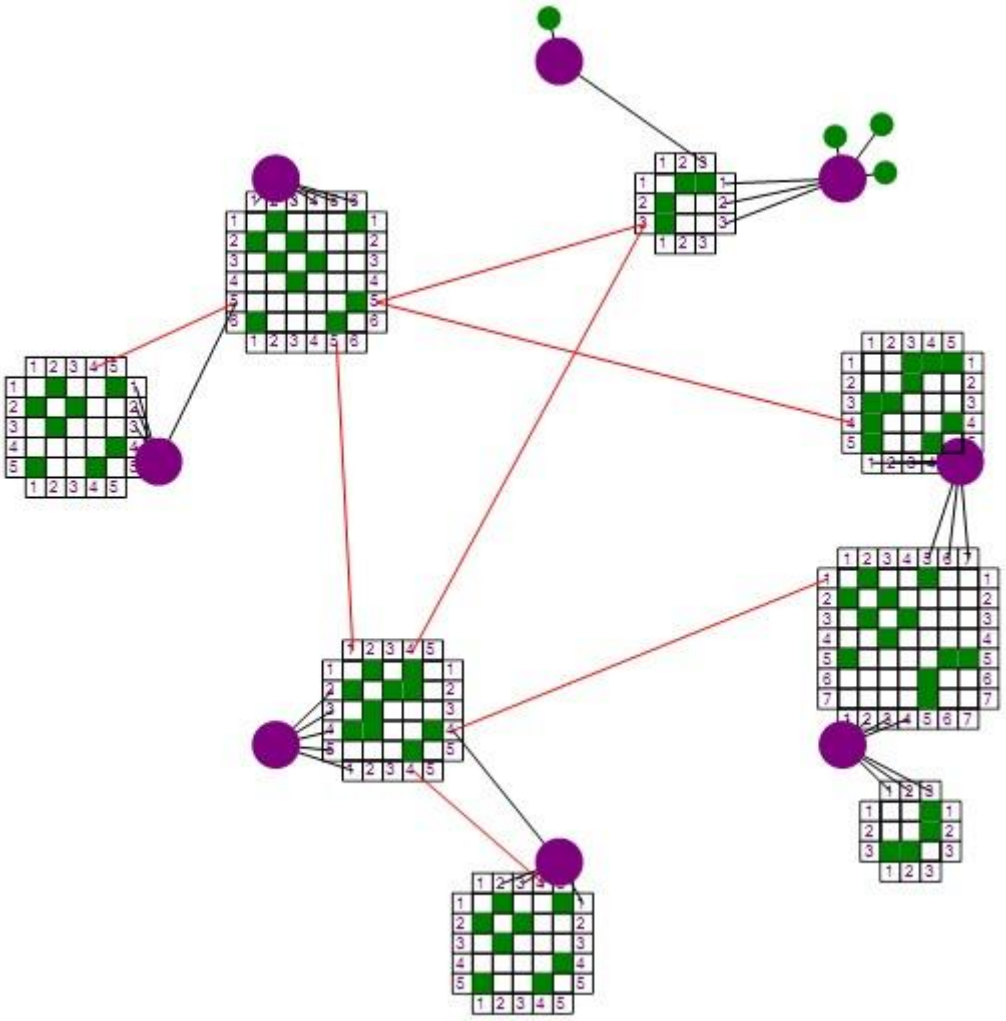


Figure 28 graph without matrix reordering

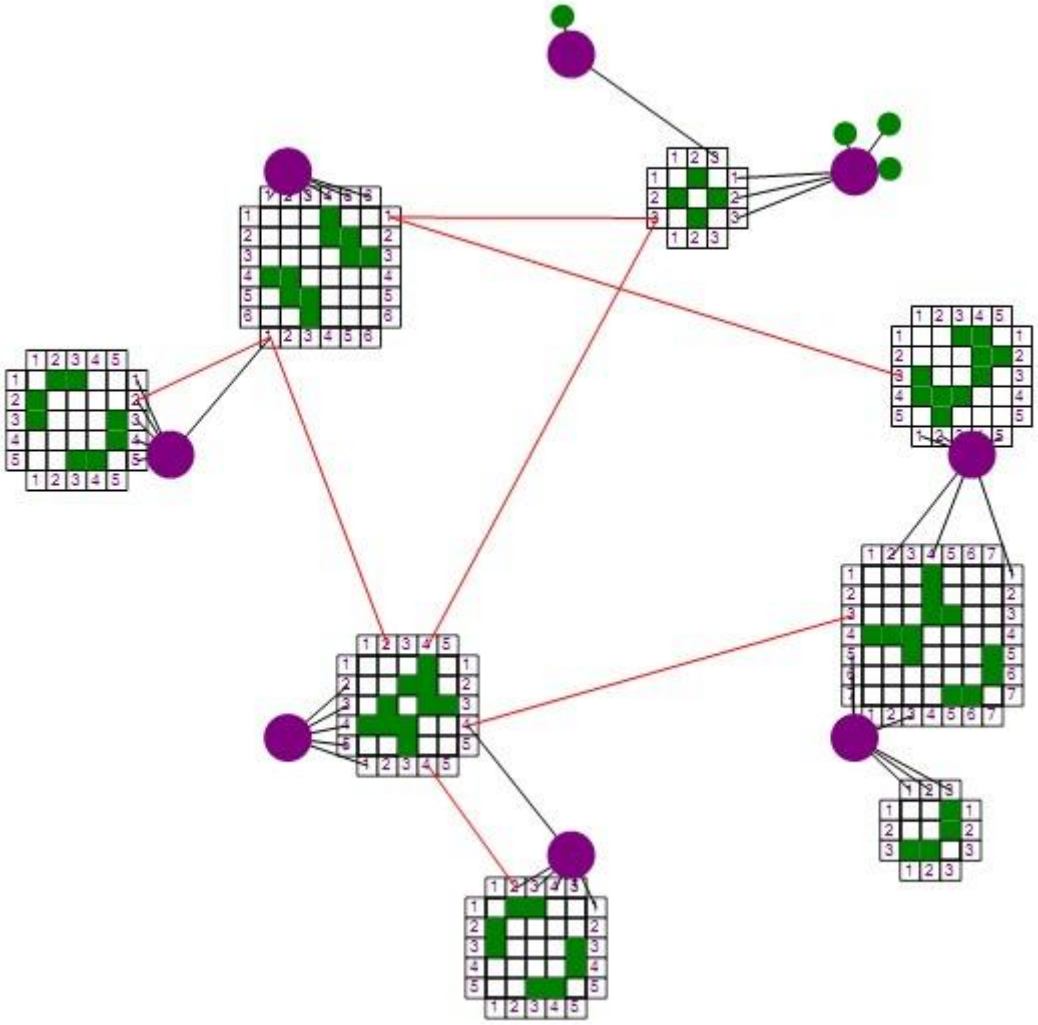


Figure 29 graph with matrix reordering

Chapter 7

Drawing Examples

In this chapter, we will discuss the characters of drawing style by showing some semi-bipartite graph data of real life including two real SNS Data and one real author-paper data. First we will show the drawing result by anchored map style, and then we will shown the drawing result of same data by proposed hybrid drawing style.

7.1 SNS Data

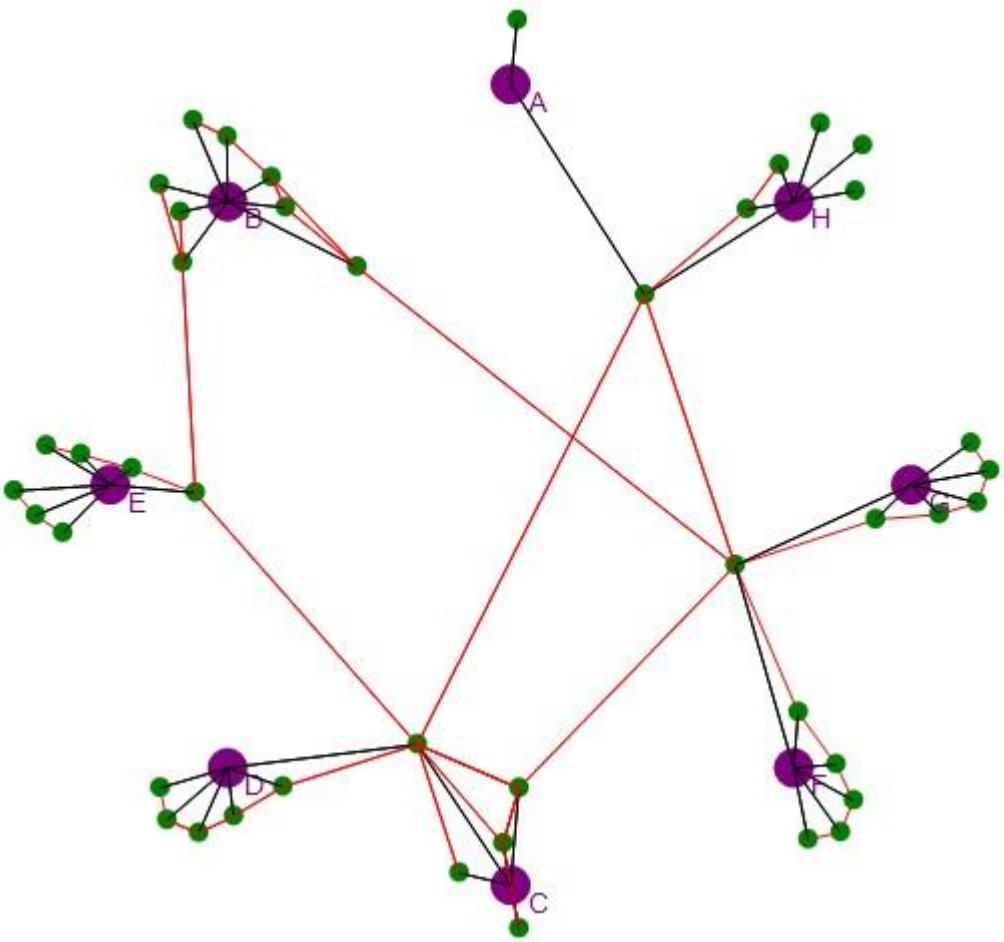


Figure 30 real SNS data 1 in anchored map style

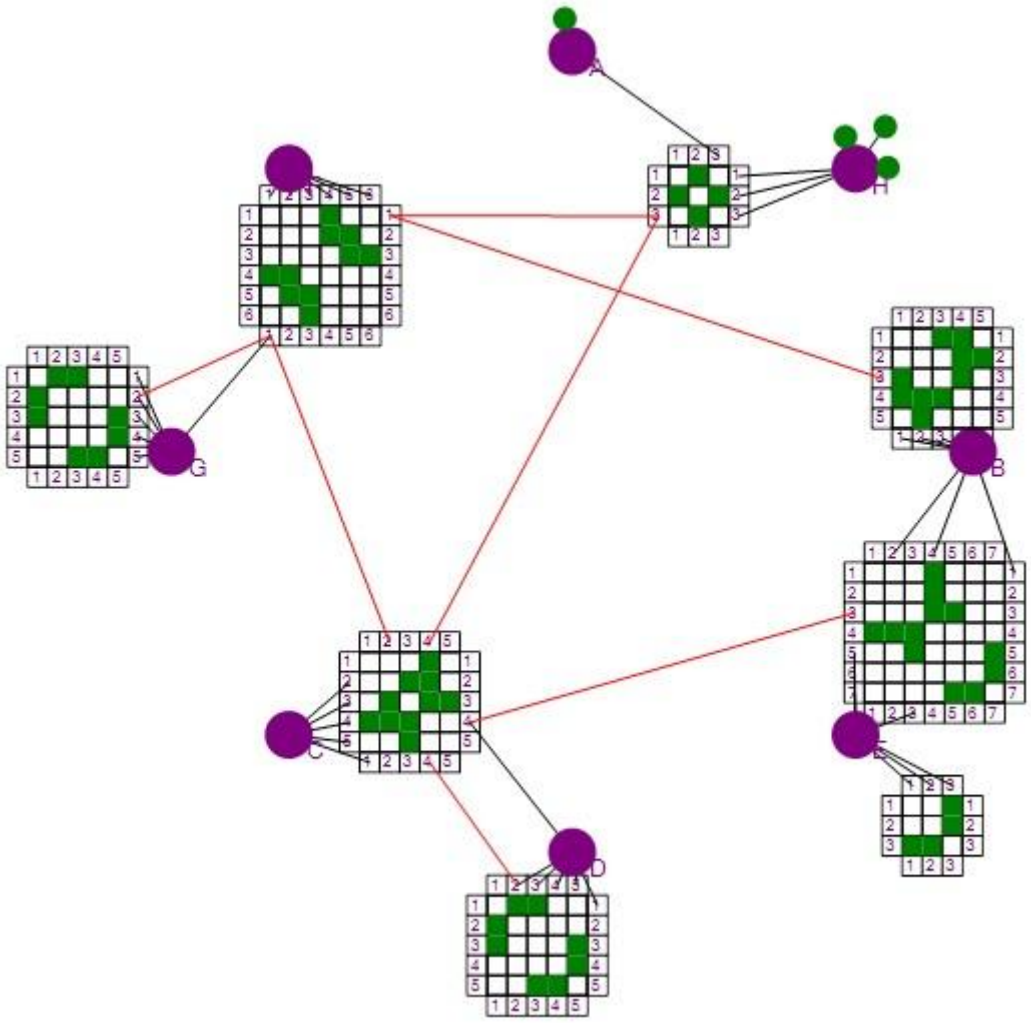


Figure 31 real SNS data 1 in proposed hybrid drawing style

A drawing result of a real social network data is showed in Figure 30 and Figure 31 where anchors represent communities and free nodes represent user. In anchored map style (Figure 30), anchors (communities) will decide the overview of the graph, and free node will be arranged to a suitable place to express its relation to related anchors and other free nodes by spring embedder model. Anchors with close relations will be fixed near to each other. In this way, the aesthetic criteria for edge crossing and length can be also satisfied.

After node clustering, users with close relations are grouped together and drawn in matrix style (Figure 31). It is obvious that users belonging to a common community are prone to have connection and grouped into same matrix with high possibility, and their relationship can be easily read by matrix representation. At the same time communities with many common user will arranged close to each other. Another obvious feature of our drawing method is that, “key person” who connected different user groups together can be easily found by reading edges between matrices.

As discussed before, matrix representation is used to improve the readability of dense graph, especially when lots of E_2 edges exist between free nodes (Figure 5). By this example we can understand that for drawing a sparse graph, the extended anchored map style may also bring a good readability, without considering the characters of node clustering it is hard to say which one is better. When drawing a dense graph (Figure 5, Figure 6) or graph where dense sub−graph exist, the proposed hybrid drawing style may bring better readability.

Figure 32 shows the drawing result of a real social network data in anchored map style, where a lot of E_2 edges exist between free nodes near anchor "7" and "8" which makes it is hard to read the relations between them.

Figure 33 shows the drawing results of the same data in the proposed hybrid drawing style where free nodes with close relations are grouped into clusters and drawn in matrix representation.

In this way, it is easy to found that almost all users in the biggest group belongs to community "1" except node "4" who belongs to community 8 but still have relations with nodes "2,3,5,6,7". And it is obvious that there is a key person node "6" who is connected with other two different user groups. At the same time, communities with many common users such as community "3" and "4" are arranged near to each other.

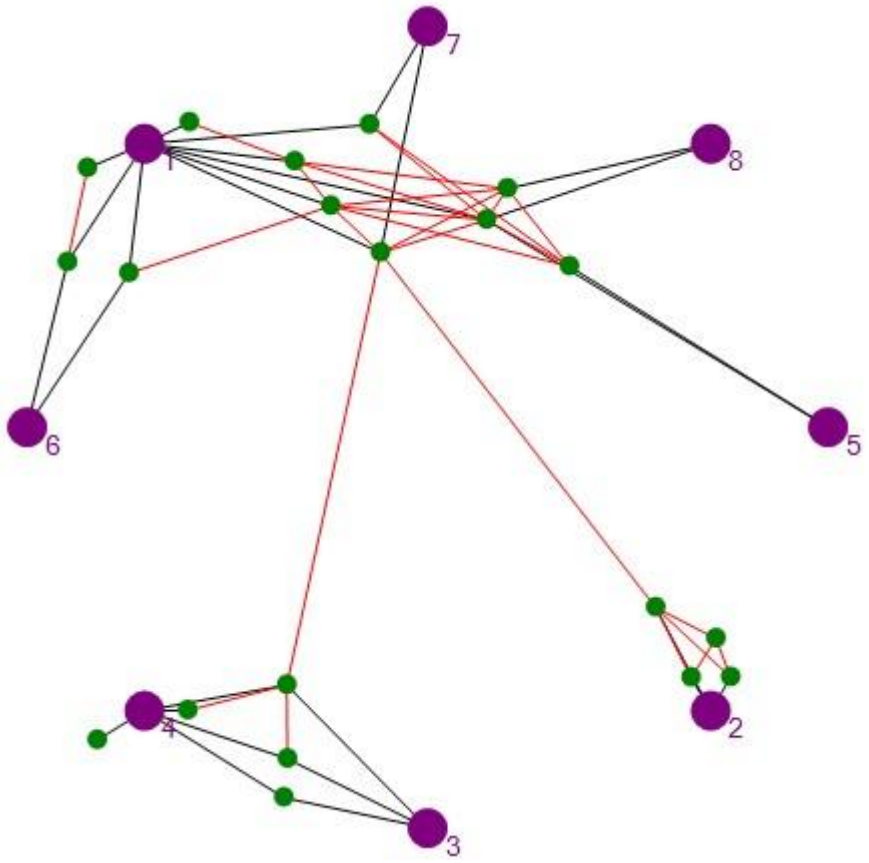


Figure 32 real SNS data 2 in anchored map style

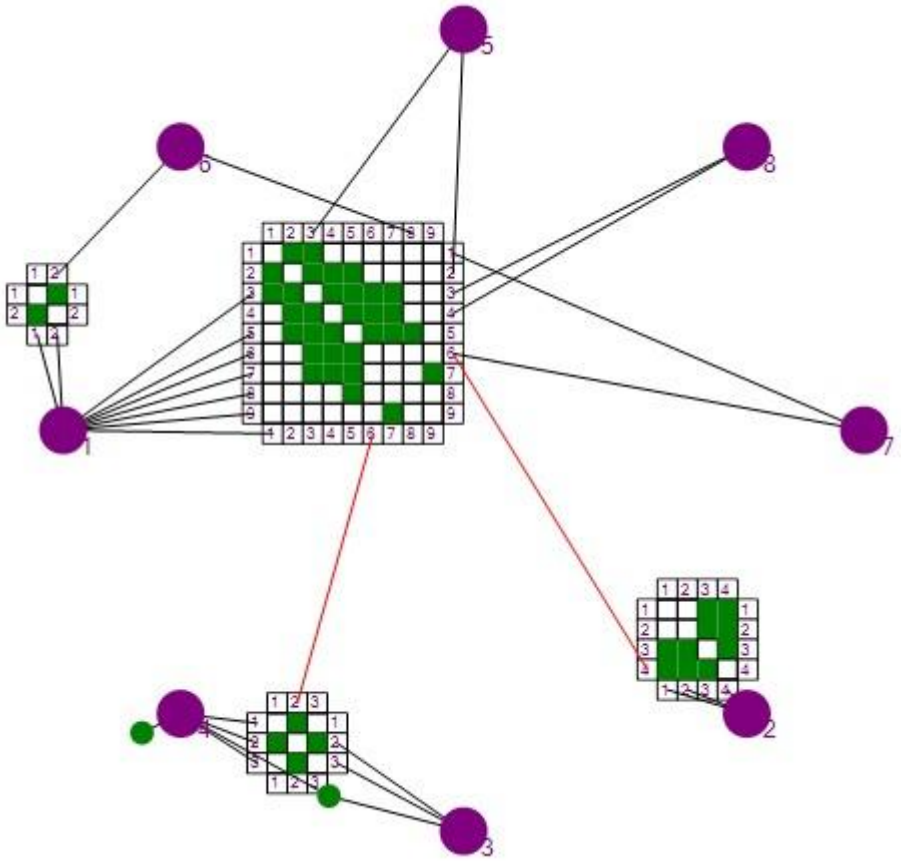


Figure 33 real SNS Data 2 in proposed hybrid drawing style

7.2 Author–Paper Data

Drawing results of an author–paper data in both anchored map style and hybrid drawing method are shown in this part.

Figure 34 shows the drawing result in anchored map style, where authors are fixed as anchors in a circumference to decide the overview and papers are displayed as free nodes. Authors with close relations (both co–author relationship and citation relationship) are set close to each other, and papers are arranged in a suitable place to reveal its relation to related authors and paper. The readability of this example is not bad, but still, by drawing it in hybrid drawing style, some potential information can be understood by the characters of node clustering.

Figure 35 shows the drawing result of author–paper data in proposed hybrid drawing style. Anchors represent authors and free nodes represent papers. It is obvious that, papers in the same research area are prone to have citation relations, so a paper cluster (matrix) can thought to be a research area. And from the drawing result, we can easily find authors who a have high possibility of focusing on the same research area, and their citation relations can be read by matrix representation.

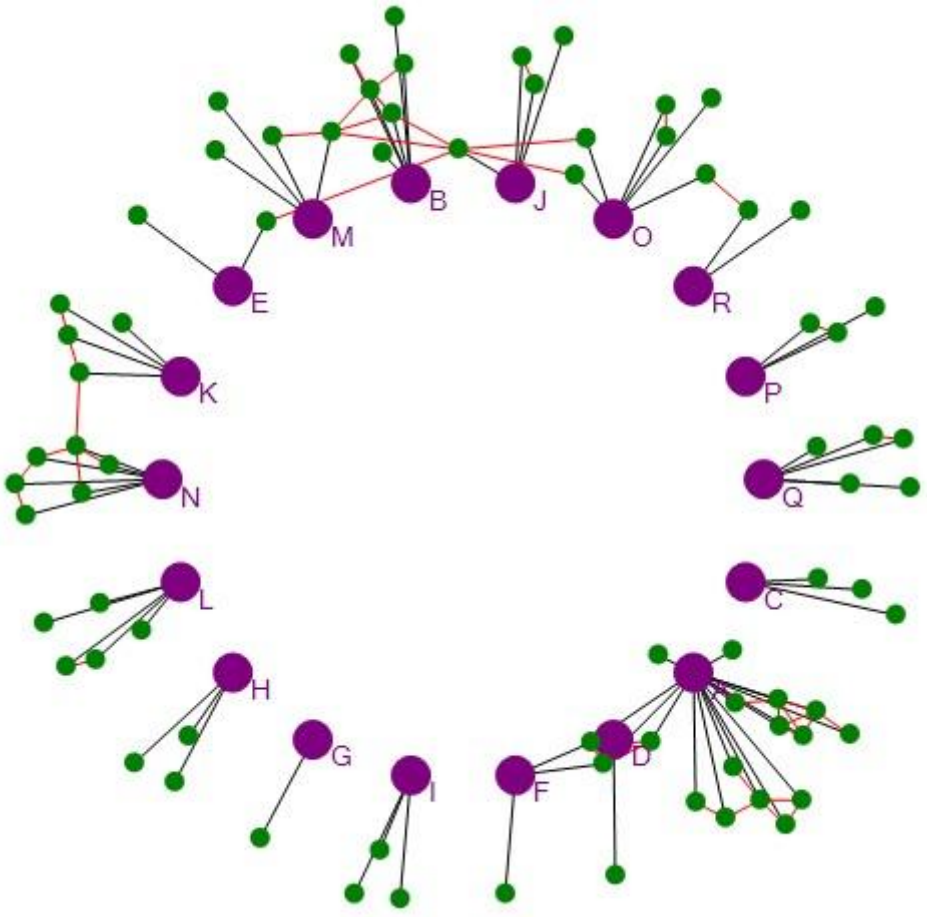


Figure 34 author–paper data in anchored map style

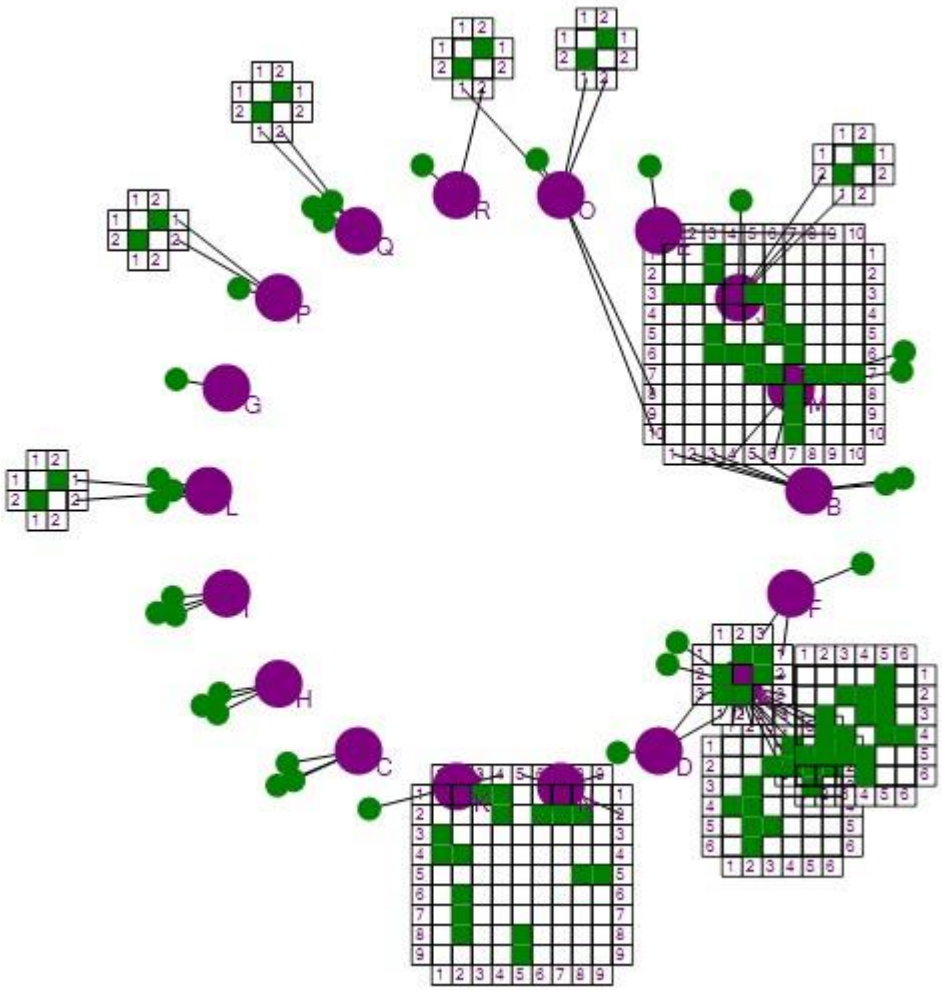


Figure 35 author–paper data in proposed hybrid drawing style

Chapter 8

Conclusion

In this paper, we represent a hybrid drawing style combined with anchored map and matrix representation. First, we extended the anchored map for drawing the semi−bipartite graphs by definition of two indexes of penalty, which has been proved to be a good index for searching good anchor order. Then, in order to improve the readability of the drawing result, we proposed a hybrid drawing style combined with the anchored map and the matrix representation. For better readability of the matrix representation, we also developed a matrix reordering algorithm based on barycenter heuristic.

To our knowledge, our research is the first drawing method which focused on visualizing semi—bipartite graph and we also want to emphasize the importance of this new graph structure since it is can be seen in many fields of real life.

By using our drawing style, the characters of the semi—bipartite graphs can be well read, at the same time, some data mining work can be also be performed based on the feature node clustering.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisor Dr. Misue Kazuo, associate professor of University of Tsukuba, for his precious advice, guidance and supervision during the course of my present study. It would not have been possible for me to complete my study without his generous training. His kind gestures will never be forgotten.

I also want to thank the other professors of our lab, Dr. Jiro Tanaka, Dr. Shin Takahashi and Dr. Buntaro Shizuki for their valuable comments and constructive suggestions.

I would also thank all the members of our lab especially the members of NAIS team who have been next to me all these years and all my friends both in China and Japan for their support and encouragement.

Bibliography

[1] J.E. Atkins, E.G. Boman, and B. Hendrickson, A Spectral Algorithm for Seriation and the Consecutive Ones Problem. SIAM J. Comput., 28(1). pp.297—310, 1998.

[2] Bertin, J. Semiologie graphique : Les diagrammes — Les reseaux — Les cartes. Editions de l'Ecole des Hautes Etudes en Sciences, Paris, France, 1967.

[3] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice—Hall, 1999.

[4] P.J. Carrington, J. Scott, and S. Wasserman. Models and Methods in Social Network Analysis. Cambridge University Press, 2005.

[5] J. Díaz, J. Petit and M. Serna, M. A survey of graph layout problems.ACM Comput. Surv., 34 (3). pp.313—356, 2002.

[6] P.Eades, “A heuristic for graph drawing”, Congressus Numerantium, vol42, pp.149—160, 1984.

[7] J. Gao, K. Misue, J. Tanaka. A Multiple—aspects Visualization Tool for Exploring Social Networks. HCI International, pp.277—286, 2009.

[8] http://www.graphviz.org

[9] E. D. Giacomo, L. Grilli, G. Liotta : Drawing Bipartite Graphs on Two Curves. In Proceedings of Symposium on Graph Drawing (GD 2006), LNCS 4372, 380—385, 2007.

[10] M. Ghoniem, J.—D Fekete and P. Castagliola. On the readability of graphs using node—link and matrix—based representations: a controlled experiment and statistical analysis. Information Visualization, 4 (2). pp.114—135, 2005.

[11] I. Herman, G. Melanc, on, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics, 6(1): pp.24—43, 2000.

[12] N.Henry, J.D.Fekete, M.J.McGuffin. NodeTrix: A Hybrid Visualization of Social Networks. InfoVis, pp.1302—1309, 2007.

[13] N. Henry and J.—D. Fekete. MatrixExplorer: a Dual—Representation Systemto Explore Social Networks. IEEE Transactions on Visualization andComputer Graphics, 12(5): pp.677—684, 2006.

[14] T. Ito, K. Misue and J. Tanaka: Sphere Anchored Map: A Visualization Technique for Bipartite Graphs in 3D. In Proceedings of 13th International Conference on Human—Computer Interaction (HCI International 2009), Human—Computer Interaction, Part II, LNCS 5611, pp.811—820, San Diego, California, USA, July 19—24, 2009.

[15] T. Ito, K. Misue and J. Tanaka: Drawing Clustered Bipartite Graphs in Multi—Circular Style. In Proceedings of 2010 14th International Conference Information Visualisation (IV2010), pp.23—28, London, United Kingdom, July 26—29, 2010.

[16] T. Itoh, C. Muelder, Kwan—Liu Ma, Jun Sese. A Hybrid Space—Filling and Force—Directed Layout Method for Visualizing Multiple—Category Graphs. IEEE Pacific Visualization Symposium, pp.121—128, 2009.

[17] K.Xu, R.Williams, SH.Hong, Q.Liu, J.Zhang. Semi—Bipartite GraphVisualization for Gene Ontology Networks. 17th International Symposium on Graph Drawing, pp. 244—255, 2009.

[18] Weimao Ke, Katy Börner, Lalitha Viswanath. Major Information Visualization Authors, Papers and Topics in the ACM Library. Proceedings of the IEEE Symposium on Information Visualization, pp. 216.1, 2004.

[19] K. Misue. Drawing Bipartite Graphs as Anchored Maps. Asia Pacific Symposium on Information Visualization, pp.169 — 177, 2006.

[20] K. Misue, “Anchored map: Graph drawing technique to support network mining,” IEICE Trans. Inf. & Syst., vol. E91—D, no. 11, pp. 2599—2606, 2008.

[21] M. E. J. Newman, The structure and function of complexnetworks. SIAM Review 45, pp.167—256, 2003.

[22] E. Mäkinen, and H. Siirtola. The Barycenter Heuristic andthe Reorderable Matrix. Informatica, 29(3):pp.357—364, 2005.

[23] M. E. J. Newman. Fast algorithm for detecting community structure in networks. Physical Review E 69, 066133, 2004.

[24] M. Newton, O. Sykora, I. Vrto : Two New Heuristics for Two—Sided Bipartite GraphDrawing. In : Proceedings of Symposium on Graph Drawing(GD 2002), LNCS 2528, pp.312—319, 2002.

[25] Rao, R. and Card, S.K. The table lens: merging graphical and symbolicrepresentations in an interactive focus + context visualization for tabularinformation in Proceedings of the SIGCHI conference on Humanfactors in computing systems: celebrating interdependence ACM Press, Boston, Massachusetts, United States, pp.318—322, 1994

[26] H. Siirtola, and E. Mäkinen, Constructing and reconstructing the reorderable matrix. Information Visualization, pp.32—48, 2005

[27] M. Spenke, C.Beilken, and T. Berlage, FOCUS: the interactive tablefor product comparison and selection in Proceedings of the 9th annualACM symposium on User interface software and technology ACMPress, Seattle, Washington, United States, pp.41—50, 1996.

[28] S. Sato, K. Misue, and J. Tanaka. Readable Representations for Large—Scale Bipartite Graphs. In Proceedings of the 12th International Conference on Knowledge—Based and Intelligent Information & Engineering Systems, pp.831—838, 2008.

[29] S. H. Strogatz, Exploring complex networks. Nature 410, pp.268—276, 2001.

[30] Kozo Sugiyama, Shojiro Tagawa and Mitsuhiko Toda: Methods for visual understanding of hierarchical system structures, IEEE Trans. on Systems, Man, and Cybernetics, vol.SMC—11, no.2, pp.109—125, 1981.

[31] L. Zheng, L. Song, P. Eades : Crossing Minimization Problems of Drawing Bipartite Graphsin Two Clusters. In : Proceedings of Asia—Pacific Symposium on Information Visualization(APVIS 2005), pp.33—38 , 2005.

[32] Q. Zhou, K. Misue, J. Tanaka. “Drawing Method Combined With Matrix Representation and Anchored Map for Semi-Bipartite Graph”. In *Transactions of Information Processing Society of Japan*, 2N-3, 2010

Appendix

This part will show the original data of correlation result which shows in 5.5. The first row shows the original graph number, the second row shows the correlation between penalty and edge–crossing, and the third row is the correlation between penalty and edge –length.

Table 1 , Table 2 shows the result of “shortest path”, and Table 3, Table 4 shows the result of “all path”.

Table 1 original data of “shortest path” method with 10 anchors

Graph Number	Edge—crossing	Edge—length
1	0.678051349	0.754610657
2	0.790203226	0.92664194
3	0.800366197	0.892971916
4	0.811477241	0.910775254
5	0.709223311	0.901997451
6	0.555584396	0.87628924
7	0.680341707	0.892757978
8	0.636464896	0.855547763
9	0.760472855	0.963328543
10	0.674814265	0.722458763
11	0.665142373	0.948474671
12	0.628384789	0.8392366
13	0.801119645	0.962968913
14	0.592648577	0.912188208
15	0.770950138	0.952219752
16	0.839793416	0.965437875
17	0.76988009	0.896481716
18	0.596521828	0.895897797
19	0.70203359	0.799903126
20	0.704879683	0.80838634
21	0.646221093	0.943700059
22	0.726867462	0.793114277
23	0.721101864	0.887439648
24	0.672013772	0.863571334
25	0.815790615	0.867726434
26	0.47625868	0.750624331
27	0.670769985	0.947327988
28	0.748267924	0.841254572

29	0.790947504	0.874627441
30	0.709694308	0.860916926
31	0.8033326	0.922399042
32	0.652477478	0.746118418
33	0.786126488	0.887408305
34	0.651636779	0.764400082
35	0.734662757	0.917671169
36	0.732228293	0.824025838
37	0.801493308	0.937675623
38	0.77127564	0.912702086
39	0.662320676	0.777840312
40	0.817257746	0.879512837
41	0.596292301	0.826021252
42	0.765584941	0.834926816
43	0.43357653	0.820809294
44	0.6702148	0.76504068
45	0.750295186	0.934120203
46	0.665110597	0.824756101
47	0.735819169	0.9190294
48	0.771573224	0.900686456
49	0.756804464	0.895913894
50	0.770913773	0.93305834
51	0.833445827	0.864222977
52	0.731175372	0.788189827
53	0.580880383	0.738843923
54	0.706840104	0.905314787
55	0.733415517	0.843835562
56	0.74853055	0.93519639
57	0.755744417	0.84043975
58	0.507473126	0.78523873
59	0.533008212	0.85246153
60	0.764045678	0.939396603
61	0.683437743	0.752172071
62	0.673865399	0.826455025
63	0.686952896	0.878890055
64	0.244352142	0.837873355
65	0.70893056	0.883690703
66	0.606356069	0.793731943
67	0.860392284	0.958791333
68	0.753718118	0.863220141
69	0.680347025	0.954430168
70	0.788853486	0.936559893
71	0.736819081	0.81118053

72	0.714984001	0.761700219
73	0.502615506	0.72075231
74	0.781971913	0.954164281
75	0.723622943	0.880951719
76	0.383312386	0.434595278
77	0.694403894	0.826097489
78	0.673589472	0.926221905
79	0.650222143	0.777093851
80	0.807071899	0.880337312
81	0.745238935	0.796878032
82	0.696252189	0.809276305
83	0.703292742	0.79716178
84	0.57814557	0.714249796
85	0.710156608	0.806818552
86	0.663377079	0.859664614
87	0.69789081	0.792068182
88	0.838357413	0.894922673
89	0.756249429	0.899579442
90	0.858635174	0.924007435
91	0.640664548	0.710548385
92	0.645017629	0.948687168
93	0.775377472	0.840806254
94	0.831283578	0.90655148
95	0.54537436	0.751916499
96	0.867025795	0.896826933
97	0.636419559	0.601325072
98	0.662601005	0.746490039
99	0.555552728	0.748699021
100	0.579583072	0.835022985

Table 2 original data of “shortest path” method with 15 anchors

Graph number	Edge—crossing	Edge—length
1	0.79441468	0.896387846
2	0.717623009	0.825702462
3	0.775840017	0.902688109
4	0.730637745	0.83550613
5	0.768511133	0.863092146
6	0.559328874	0.584290074
7	0.748323005	0.856870715
8	0.77291464	0.95068606
9	0.619891681	0.697019445
10	0.780118561	0.898753417

11	0.709898227	0.863435772
12	0.803990095	0.892156424
13	0.785386574	0.919170433
14	0.832167453	0.927344354
15	0.808538852	0.909513645
16	0.684734821	0.732574238
17	0.736916769	0.879535542
18	0.650868862	0.795718798
19	0.672823379	0.801168013
20	0.725906849	0.765486524
21	0.719861982	0.820542414
22	0.573919309	0.719613457
23	0.800628376	0.938709472
24	0.794984404	0.892589335
25	0.827944235	0.923903127
26	0.765994762	0.85253139
27	0.754245437	0.853174612
28	0.688975122	0.785690927
29	0.747048408	0.86706537
30	0.660023407	0.822206349
31	0.6709934	0.788533054
32	0.694418453	0.844994866
33	0.777388479	0.87893366
34	0.851436523	0.905994421
35	0.815869187	0.93610727
36	0.805002528	0.875795853
37	0.690198988	0.90040276
38	0.716298357	0.800707157
39	0.603463771	0.728377359
40	0.649432311	0.780330635
41	0.70276412	0.801188417
42	0.809182234	0.872372858
43	0.809494493	0.926103836
44	0.656150923	0.750440275
45	0.663874791	0.759073035
46	0.726377547	0.869913595
47	0.668610158	0.835236855
48	0.787788898	0.898262449
49	0.68388357	0.823107293
50	0.803735518	0.916314117
51	0.744712749	0.897326173
52	0.739677109	0.865516997
53	0.812883474	0.944854202

54	0.771969732	0.838064014
55	0.723275106	0.81890675
56	0.610866392	0.839962802
57	0.650905414	0.766227916
58	0.741315229	0.852372262
59	0.752193724	0.850922433
60	0.693362969	0.796718512
61	0.771302025	0.891710376
62	0.764373465	0.882348762
63	0.824981446	0.902440029
64	0.727658474	0.894567898
65	0.717047434	0.784091386
66	0.702103897	0.789162672
67	0.686401432	0.802790026
68	0.665724256	0.746549935
69	0.692015073	0.850392912
70	0.451653944	0.617825108
71	0.643491382	0.801807923
72	0.694960155	0.73730156
73	0.569299147	0.760590469
74	0.630823245	0.688687019
75	0.770898882	0.841757311
76	0.683312618	0.768982536
77	0.671697715	0.772054664
78	0.564369115	0.516072513
79	0.729520803	0.909395782
80	0.742470145	0.90807058
81	0.744041031	0.847109463
82	0.746438714	0.930318753
83	0.614412989	0.804113635
84	0.621152768	0.834181844
85	0.786987763	0.898872669
86	0.809058055	0.893420213
87	0.697435361	0.773255581
88	0.61048595	0.706271517
89	0.750472897	0.822421151
90	0.721346876	0.790365518
91	0.763292392	0.819178908
92	0.731379861	0.826707682
93	0.762290846	0.816363724
94	0.711879497	0.865278874
95	0.691321583	0.794137922
96	0.731972456	0.798619051

97	0.71335237	0.809133424
98	0.786082947	0.865834881
99	0.757641826	0.855735408
100	0.683975386	0.748717832

Table 3 original data of “all paths” method with 10 anchors

Graph number	Edge—crossing	Edge—length
1	0.664126761	0.848109609
2	0.833603953	0.961320232
3	0.841593038	0.930448247
4	0.84000802	0.937327397
5	0.538025927	0.826401327
6	0.493683558	0.866047047
7	0.641229904	0.942918514
8	0.714985665	0.933150088
9	0.785778805	0.973198022
10	0.692639914	0.864433816
11	0.520089757	0.943757033
12	0.42677108	0.852797274
13	0.790831146	0.967250352
14	0.686526392	0.921008348
15	0.768505424	0.954404201
16	0.832173653	0.959663732
17	0.737892203	0.935451911
18	0.567843865	0.92615098
19	0.723541093	0.886842206
20	0.623579192	0.904709457
21	0.494319312	0.84408989
22	0.811127767	0.897571111
23	0.545703515	0.9059786
24	0.693076445	0.856541884
25	0.873509477	0.94101878
26	0.531250672	0.847379336
27	0.665647302	0.951235443
28	0.77296794	0.917990342
29	0.821385936	0.947334692
30	0.689115895	0.888132752
31	0.832045413	0.952696539
32	0.403943904	0.764837764
33	0.83307367	0.940536938
34	0.505447439	0.866372694
35	0.632842526	0.899450108

36	0.818314518	0.925735499
37	0.856000508	0.965031778
38	0.736733602	0.906820589
39	0.678195582	0.883444779
40	0.806231912	0.908967573
41	0.346240082	0.812845744
42	0.785050716	0.877102507
43	0.350009515	0.925818248
44	0.719430864	0.864207665
45	0.789013975	0.944204304
46	0.730234887	0.908192683
47	0.683136293	0.895573896
48	0.713423413	0.875035813
49	0.777541692	0.947364882
50	0.780315512	0.934225485
51	0.833670633	0.867740649
52	0.752935089	0.895455541
53	0.666057566	0.914992759
54	0.709521372	0.935586585
55	0.753360324	0.850255717
56	0.756123998	0.938408652
57	0.692079707	0.878078887
58	0.560386071	0.877194785
59	0.555291085	0.935444529
60	0.783248632	0.943582462
61	0.661273278	0.795387523
62	0.787608405	0.898492752
63	0.747265368	0.939550623
64	0.216344946	0.858643268
65	0.627640609	0.908652694
66	0.614243691	0.915523911
67	0.8338543	0.937059525
68	0.77258037	0.872631961
69	0.64944883	0.925794449
70	0.794588643	0.937438564
71	0.694280291	0.844224242
72	0.603467317	0.699119309
73	0.21779353	0.719747085
74	0.795351485	0.942882636
75	0.512348487	0.872946087
76	0.580443747	0.771664534
77	0.770957909	0.909964492
78	0.628419287	0.915551492

79	0.634861171	0.848331394
80	0.816952626	0.884113473
81	0.765286093	0.894693618
82	0.614843576	0.866400632
83	0.743284783	0.928782189
84	0.529321048	0.757866575
85	0.782610276	0.851717251
86	0.68613947	0.895654335
87	0.578010351	0.692054555
88	0.576310165	0.774544235
89	0.789806084	0.913893672
90	0.812611607	0.884168503
91	0.403746355	0.710833671
92	0.658647452	0.962921009
93	0.71569183	0.818201106
94	0.78862048	0.873005219
95	0.090979622	0.584747892
96	0.84237775	0.914361101
97	0.245928656	0.470124984
98	0.721291995	0.861931187
99	0.573390623	0.788398729
100	0.629934685	0.904083317

Table 4 original data of “all paths” method with 15 anchors

Graph number	Edge—crossing	Edge—length
1	0.721592674	0.844085342
2	0.658274343	0.84744768
3	0.675869718	0.849057041
4	0.592128265	0.74805479
5	0.77944701	0.887490923
6	0.238748872	0.477980249
7	0.677605246	0.89390789
8	0.809181577	0.96560259
9	0.382373244	0.579600645
10	0.774382246	0.921544454
11	0.681238824	0.872369326
12	0.798482449	0.912487933
13	0.773274753	0.926739971
14	0.878598428	0.94860905
15	0.732062535	0.882906843
16	0.628205182	0.682068804
17	0.514710441	0.727850419

18	0.737388395	0.906485637
19	0.367790464	0.64691001
20	0.604067153	0.707043834
21	0.833605025	0.91695339
22	0.665944938	0.760965284
23	0.792154934	0.926319378
24	0.788375701	0.863598903
25	0.782651532	0.875197999
26	0.753874854	0.862074004
27	0.79178248	0.946690863
28	0.407631368	0.590760462
29	0.746467839	0.881869324
30	0.482951562	0.802786457
31	0.3834656	0.676372891
32	0.642740263	0.838491594
33	0.731113723	0.859051092
34	0.777199071	0.83936306
35	0.829694999	0.938584261
36	0.613164801	0.799984154
37	0.712511548	0.910912076
38	0.626151414	0.691110857
39	0.383802926	0.682989762
40	0.324181118	0.768151339
41	0.68256582	0.817198632
42	0.795442764	0.846218236
43	0.795499654	0.930025952
44	0.510225956	0.68644824
45	0.13005414	0.578240861
46	0.79535719	0.860895594
47	0.597023856	0.857080012
48	0.67090451	0.828905773
49	0.495420974	0.758543163
50	0.788860946	0.905713555
51	0.745446249	0.937074406
52	0.561825903	0.734180203
53	0.800387644	0.916877264
54	0.815798729	0.888751061
55	0.558747878	0.687281484
56	0.335878123	0.690271365
57	0.48617384	0.698075726
58	0.67824684	0.857899262
59	0.692931307	0.838410539
60	0.83797675	0.921846153

61	0.692548691	0.819835718
62	0.822889813	0.934972282
63	0.63337801	0.796891231
64	0.649616014	0.814001743
65	0.773521134	0.833519363
66	0.438647443	0.579399935
67	0.643807376	0.774004947
68	0.702222767	0.870542106
69	0.531864924	0.710918541
70	0.539005294	0.714021983
71	0.602144143	0.827326813
72	0.288645972	0.619335431
73	0.51517039	0.819962063
74	0.431519329	0.627155158
75	0.76858594	0.871956762
76	0.661537986	0.824535971
77	0.572679628	0.669403973
78	0.167097608	0.435053448
79	0.741013578	0.893115851
80	0.654703401	0.858466785
81	0.706487756	0.831125821
82	0.649091293	0.810557707
83	0.568068852	0.819891277
84	0.716109748	0.91257036
85	0.516949554	0.781418938
86	0.71903599	0.800079159
87	0.578223921	0.707874264
88	0.456750434	0.66699031
89	0.593724708	0.757497569
90	0.359965884	0.502428732
91	0.563592446	0.684386907
92	0.686561509	0.841160598
93	0.726697954	0.836222031
94	0.549078912	0.656785363
95	0.700791423	0.876193596
96	0.800620393	0.890420146
97	0.739859206	0.798761141
98	0.69862668	0.89909744
99	0.736268537	0.861211442
100	0.713321991	0.780908588