Drawing Bipartite Graphs as Anchored Maps

Kazuo Misue

Graduate School of Systems and Information Engineering
University of Tsukuba
1-1-1 Tennoudai, Tsukuba, 305-8573 Japan

misue@cs.tsukuba.ac.jp

Abstract

A method of drawing anchored maps for bipartite graphs is presented. Suppose that the node set of a bipartite graph is divided into set A and set B. On an anchored map of the bipartite graph, the nodes in A, which are called "anchor nodes," are arranged on the circumference, and the nodes in B, which are called "free nodes," are arranged at suitable positions in relation to the adjacent anchor nodes. This article describes aesthetic criteria that are employed according to the purpose of drawing anchored maps and a method to arrange the anchor nodes so that they satisfy the criteria. The effectiveness of the proposed method is discussed in terms of the aesthetics of drawing results.

Keywords: graph drawing, anchored map, bipartite graph

1 Introduction

Bipartite graphs can be found in various fields. Examples include graphs that show relationships between documents and the words included in them, graphs that show relationships between customers and goods they bought, and graphs that show relationships between communities and their members.

Such relationships are often used to compute similarities. For example, when customer A and customer B have bought a lot of the same goods, we may think that A and B share similar properties. On the other hand, when a lot of customers bought goods X together with goods Y, we may think X and Y have some similarities. Such similarities are used for information retrieval, E-commerce recommendations, and so on.

The purpose of this research is visualization of the relations represented by bipartite graphs. In particular, its target is to facilitate an understanding of the node clusters' meaning and the relationships among the clusters. In the examples mentioned above, we expect that bipartite graphs can help us to understand the relational structures such as the scale of clusters of similar goods, the relationships among clusters of goods, and the relationships among clusters of goods and customers.

We developed a drawing technique for expressing bipartite graphs called "anchored maps." It has two kinds of node: "anchors" and "free nodes." The anchors are arranged on the circumference at equal intervals, and the free nodes are arranged at suitable positions in relation to the adjacent anchor nodes. The problem is how to decide the order of anchor nodes on the circumference and the positions of free nodes. We do not need to worry about the routing of edges because we use straight-line drawing.

In this article, we describe the technique for drawing anchored maps. We explain the aesthetic criteria that are employed according to the map's purpose and a method to arrange anchor nodes to fulfill the criteria. We also show the technique's effectiveness in an aesthetic evaluation of its drawing results.

2 Anchored maps of bipartite graphs

2.1 Bipartite graphs

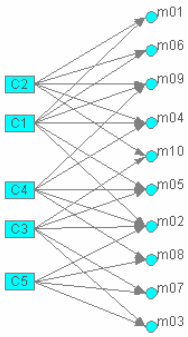
Suppose that $G = (A \cup B, E)$ is a bipartite graph. A and B are finite sets of vertices, and A and B are disjoint. E is a finite set of edges, and E is a subset of $A \times B$.

2.2 Expression styles of bipartite graphs

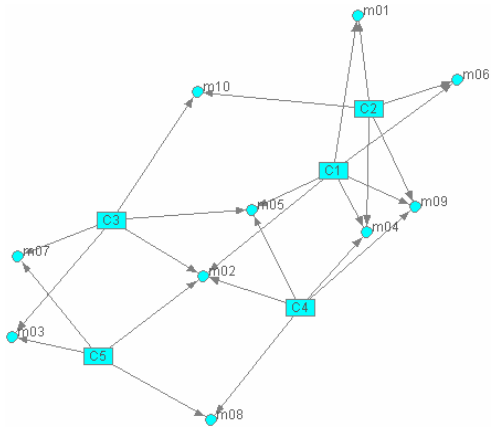
Figure 1 shows various styles of expressing bipartite graphs. Virtual data is used as an example. Suppose that there is a bipartite graph that represents relationships between five communities and ten members. Figure 1(a) shows the matrix representation. Each row corresponds to a community, and each column corresponds to a member. A "1" in a cell indicates that a member belongs to a community; for example, member m01 belongs to communities C1 and C2. This matrix cannot intuitively show relational structures because it is not visual. Figure 1(b) shows a two-layer graphical representation of the same bipartite graph. The five nodes on the left-hand side represent communities, and the ten nodes on the right-hand side represent its members. A line between a community and a member shows that the member belongs to the community. Figure 1(c) shows the same graph as laid out by using spring embedding (Eades 1984). While Figure 1(b) gives the impression that the arrangement is more orderly than (c)'s, it doesn't emphasize the relations between communities and members. On the other hand, while (c) illustrates these relationships, its distinction between communities and members is not clear because the two kinds of node exist together.

	C1	C2	C3	C4	C5
m01	1	1			
m02	1			1	1
m03			1		1
m04	1	1		1	
m05	1		1	1	
m06	1	1			
m07			1		1
m08				1	1
m09	1	1		1	
m10		1	1		

(a) Matrix style



(b) Two-layer style



(c) Spring-embedded style

Figure 1: Various styles of representing bipartite graphs

2.3 Anchored Maps

An anchored map restricts some nodes to certain positions, but leaves other nodes so they may be arranged freely. The restricted nodes are fixed like anchors, hence the terms “anchors” and “free nodes.”

There are many variations in the restriction of the anchors. In the simplest one, each anchor is fixed to a coordinate. It is also possible to restrict anchors to a certain curve, in a certain area, and so on. Figure 2 shows the same bipartite graph shown in Figure 1 as an anchored map. The communities are arranged on the vertices of a pentagon, and members are arranged at a suitable position to represent their relationship to the communities. The distinction of the nodes is clearer than in Figure 1(c), and it is easy to see the relations between members.

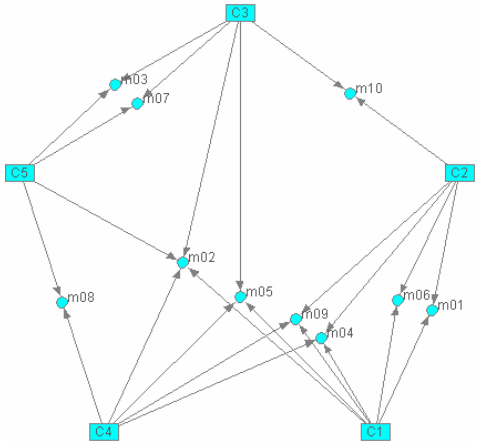


Figure 2: Anchored-map style

2.4 Drawing Convention

The drawing convention of anchored maps is as follows:

(Arrangement convention for the nodes)

- Compound coordinate system: For a bipartite graph $G = (A \cup B, E)$, the elements of set A (the anchors) depend on a circular coordinate system; they are arranged on the circumference at equal intervals (in other words, they are arranged on the vertices of a regular polygon if $|A| \geq 3$), while the elements of set B (the free nodes) are independent of the coordinate system.

(Routing convention for the edges)

- Straight line wiring: The adjacent nodes are connected by a straight line.
- Edges are independent of the coordinate system.

2.5 Features of anchored maps

The features of the anchored maps depend on how one restricts the anchor positions and how one arranges the free nodes.

For example, when anchor nodes are separated by enough distance and the free nodes have spring embedding, the anchor nodes connected to a common free node pull it in different directions. By the nature of the spring embedding, the free node will “move” to an appropriate position that expresses its relation to the connected anchor nodes.

Misue and Watanabe showed an example of expressing the relations between concepts extracted from a large volume of textual data in maps with three anchors (Misue & Watanabe 1999). Compared with simple spring embedding, the anchored maps were able to clarify the concepts that were worth attention, and thereby facilitate an easier analysis of large textual data.

2.6 Purpose in drawing anchored maps

The purpose of a bipartite graph visualization obviously varies with the application. However, the essential reasons for using anchored maps are as follows.

(a1) The positions of free nodes are understandable in relation to the anchors.

Free nodes should be arranged close to anchors. Two or more anchors connected to the free node pull it towards them. This is advantageous for understanding which anchors are related to which free node.

(a2) Clusters of free nodes can be discerned based on their relation to the anchors.

The free nodes are divided into clusters by the connective relationships with the anchors (subsets of anchors which are connected to the free nodes). This illustrates what kinds of cluster exist and which anchor subset is responsible for constructing each cluster.

3 Drawing anchored maps

3.1 Aesthetic criteria

We employed the following aesthetic criteria for our anchored map drawing method. Note that some of these criteria are employed in many drawing methods. We also judged them to be important in consideration of the adaptability to the purpose of drawing of anchored maps.

(e1) Nodes connected to each other are laid out as closely as possible (minimize the total length of edges).

(e2) The number of crossings among edges is minimized.

(e3) Anchors connected to common free nodes are laid out as closely as possible.

(e4) Free nodes belonging to the same cluster are laid out as closely as possible (free nodes do not belong to the same cluster are not laid out closely).

3.2 Drawing procedure

The map is laid out in two steps:

(Step 1) Arrange anchors on the circumference at equal intervals.

(Step 2) Fix the anchors, and arrange the free nodes at the positions in which their relationships to the anchors are appropriately expressed.

In Step 1, the size (i.e., radius) of the circumference is decided, and the order of the anchors on the circumference is decided. The length of the circumference doesn't influence the quality of the layout; it influences only the size of the drawing. Here, the circumference is decided according to the size of the drawing area (i.e., window). On the other hand, the order of the anchors has a large influence on the quality of the layout. The next section describes how to decide the order of the anchors.

In Step 2, the positions (i.e., coordinates) of the free nodes are decided. Spring embedding with restrictions is used to find the positions. That is to say, in every iteration of spring embedding, the anchors are fixed and their positions are not changed, and only the free nodes are moved toward stable positions.

3.3 How to decide the order of anchor nodes

The distance between adjacent nodes (length of the edges) and the number of edge crossings are essentially influenced by the order of the anchor nodes. However, these features are not deterministically decided from only the anchor node order. The spring embedding in Step 2 and the initial positions of the free nodes also influence the distance. However, spring embedding has negligible influence compared with the influence of the anchor order.

Thus, the order of the anchors is the most critical problem. We can evaluate the goodness of a certain order only after the spring embedding has been processed for a certain anchor arrangement, but to do this for all the possible candidate orders would require too much computing time. Thus, we need an alternative index that is computable in a deterministic way and with a low cost.

We propose to use the distance along the circumference between anchors connected to the same free nodes as such an index.

Although the computing cost of this index is not very low, we use it as the initial choice because it can be computed deterministically.

3.3.1 Preparation

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

Suppose that $p(a)$ is the position of anchor a.

3.3.2 Gap

The "clockwise distance" between the i-th vertex and the j-th vertex is the number of vertices we meet when we trace the vertices of the M-gon from the i-th to the j-th clockwise. The **clockwise distance** l_M is defined as

$$l_M(i, j) = (j - i + M) \bmod M \quad (1)$$

Next, we introduce a "gap" for every anchor connected to a certain free node. Suppose that f is a free node and $A' = \{a_1, a_2, \cdots, a_k\}$ is a set of anchors connected to f. Moreover, suppose that $a_1, \cdots, a_k$ have been sorted by the numbers of vertices of the M-gon. In other words, $p(a_s) < p(a_t)$ if $s < t$. The gap for each anchor is the clockwise distance between one node and the next node (the next of a_k is a_1). The **gap** $g(a_s)$ of node a_s is defined as

$$g(a_s) = l_M(p(a_s), p(a_{(s+1) \bmod M})) \quad (2)$$

3.3.3 Penalty

When k anchors are connected with a certain free node, a sequence of k gaps is obtained. Removing the maximum gap from the sequence leaves k-1 smaller gaps. The "penalty" is the sum of powers of k-1 gaps. A small penalty indicates that the anchors connected to the same free node should be arranged close to each other. Therefore, the penalty could be used as an index of how to decide the order of the anchor nodes.

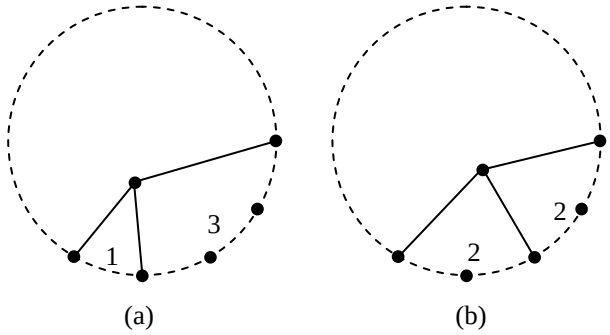


Figure 3: Effect of the parameter *q* in the penalty

Suppose that f is a free node and $g_1, g_2, \ldots, g_{k-1}$ is the remainder sequence that excludes the maximum gap from the gap sequence of the anchors connected to f. The **penalty** $p(f)$ of free node f is expressed as

$$p(f) = \sum_{i=1}^{k-1} g_i^q \tag{3}$$

Figure 3 illustrates the reason why we employed the power of the gaps with the parameter q. We introduced q to judge which of (a) or (b) is better when there is a free node in Figure 3. When $q = 1$, the penalty $p = 1^1 + 3^1 = 4$ in the case of (a), and $p = 2^1 + 2^1 = 4$ in the case of (b). The penalty is the same value in either case; it is a useless index for deciding which arrangement is better. To take account of the balance and symmetry and to prefer the case of (b) we should have $q > 1$ (for example, $q = 2$). When $q = 2$, $p = 1^2 + 3^2 = 10$ in the case of (a) and $p = 2^2 + 2^2 = 8$ in the case of (b); hence, (b) is chosen for the graph.

3.3.4 Searching for the optimal order

It is necessary to search for the anchor order that minimizes the penalty.

A simple technique is to compute the penalties for all the possible orders (isomorphism cause by rotation and mirroring is negligible), and then finds the order with the minimum penalty. This technique gives the optimal solution for the penalty but its computational cost is very large. Moreover, we should recall that the penalty is an alternative index. There is no guarantee that the optimal solution for the penalty is the optimal solution for the aesthetic criteria.

We created a quasi-optimal solution for the penalty and devised a technique with a lower computational cost. The technique begins with a random anchor order, and exchanges two anchor nodes while comparing the penalty before and after the exchange. It continues to choose pairs of anchor nodes and it exchanges them if the penalty decreases by doing so. The technique is as follows.

(a) First, we sequentially exchange adjoining nodes clockwise in a manner like bubble sorting. However, unlike bubble sorting, there is no terminal element of a list because the nodes are on a circumference. If the penalty does not change after the exchange procedure makes a complete circuit around the circumference, it stops. This procedure must stop because the penalty does not keep decreasing indefinitely. However, if we use only this condition for stopping, the solution tends to fall into a local minimum, which may not yield good results.

(b) Next, we vary the distance between the compared nodes from a large value to a small one. We start the comparison by choosing nodes far from each other, then gradually shorten the distance between nodes to be compared, and finally compare adjoined nodes at the end. For each distance, the comparison stops when the penalty does not change. In this way, it becomes unlikely that the solution will fall into a local minimum solution.

(c) To improve the efficiency of (b) at the end, the distance between nodes is made to be half the node distance of (b).

While the frequency of loop of every distance between compared nodes is $O(|A|^2)$ in (b), we expect that it becomes $O(|A| \log |A|)$ in (c). The procedure is listed as Algorithm 1 (the vertices run from 0 to $|A|-1$ in the algorithm).

Algorithm 1

```
int p0 = current penalty;
int d = |A| / 2;
while (d > 0) {
    do {
        boolean c = false;
        for (int i = 0; i < |A|; i ++) {
            int j = (i + d) mod |A|;
            swap i-th node and j-th node;
            int p1 = current penalty;
            if (p1 < p0) {
                p0 = p1;
                c = true;
            } else {
                swap i-th node and j-th node;
            }
        }
    } while (c);
    d = d / 2;
}
```

4 Evaluation concerning aesthetic criteria

We implemented the technique described in Section 3 in Java and performed an experiment to evaluate its effect in terms of aesthetic criteria.

4.1 Evaluation Items

We wanted to assess

(a) the validity of the penalty and

(b) the effectiveness of the penalty minimization algorithm.

Regarding (a), we obtained layouts of many variations for a number of graph instances and examined the correlations between the penalty and average edge length and between the penalty and the number of edge crossings for each

graph. We judged the penalty to be valid as an alternative index if there was a certain correlation between the penalty and these values.

Regarding (b), we assumed that the penalty from (a) is valid. We examined how well the solutions obtained by the algorithm compared with the optimal solutions. We judged the algorithm to be effective if the quasi-optimal solution obtained by the algorithm was closer than a random order to the optimal solution.

4.2 Outline of the evaluation

Besides the proposed algorithm, we implemented two programs in Java; one obtained the optimal order concerning the penalty, and the other obtained a random order. We generated six kinds of bipartite graph. The three programs obtained a total of 1000 kinds of layout for each of the six graphs. The 1000 layouts had the following properties.

1. One optimal arrangement of the penalty
2. One Quasi-optimal arrangement of the penalty (obtained by Algorithm 1)
3. 998 random arrangements

Although the random arrangements do not need the penalty computation, we computed the penalties of all layouts as well as the correlations between the penalties and total edge lengths and the correlations between the penalties and the number of edge crossings.

Table 1 shows the size of the six bipartite graphs used for the evaluation. Every graph had 10 anchors. Note that our technique does not need to be used when there are only a few anchors because it would not take a long time to compute all possible orders in this case. On the other hand, when the anchor nodes exceed 10 in number, it becomes difficult to obtain optimal solutions for all graphs.

Table 1: Size of the graphs used for the evaluation

	$\|A\|$	$\|B\|$	$\|E\|$
G1	10	16	65
G2	10	19	84
G3	10	40	182
G4	10	48	251
G5	10	98	473
G6	10	99	512

4.3 How to generate random graphs

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. There can be 2^n kinds of free node when paying attention to the connecting pattern to the anchors (isolated free nodes, which are not connected any anchors, are also included in this number). The arrangement of free nodes connected to at most one anchor is not influenced by the order of the anchor nodes. Thus, we generate only random graphs whose free nodes have at least degree 2. We choose free nodes of degree 2 or more with probability p. Two graphs generated with the same n and p might have different numbers of edges because p is not dependent on the connecting patterns to the anchor nodes.

The values of p for graphs G1, …, G6 are respectively 0.02, 0.02, 0.05, 0.05, 0.1, and 0.1. We assumed that p was at most 0.1 because the density of the edges rises too much when there are more free nodes, and readable layouts cannot be expected for such anchored maps.

4.4 Evaluation result

We obtained 1000 sets of penalty, average edge length, and edge crossings for each of the six graphs. Table 2 shows the correlations calculated from them, and all of them are 0.8 or more. The penalty thus seems to be useful as an alternative index as long as we can see the correlations between it and the average edge length and between it and the number of edge crossings. We also found that scatter charts could represent relations between the penalty and the average edge length, and between the penalty and the number of edge crossings. Figure 4 shows a scatter chart, which represents the relation between the penalty and the number of edge crossings in G4. Certainly, a general trend is that the number of edge crossings is small in a layout whose penalty is small. However, we can see that some cases with few edge crossings exist and their penalties are not the minimum values.

Table 2: Correlations of the penalty with the average edge length and the number of edge crossings

	Correlation between penalty and average edge length	Correlation between penalty and number of edge crossings
G1	0.884	0.828
G2	0.872	0.829
G3	0.893	0.878
G4	0.831	0.816
G5	0.904	0.893
G6	0.875	0.863

Next, we ranked the 1000 kinds of layout according to the aesthetic criteria for each graph. Table 3 shows the ranks of the optimal layout and the quasi-optimal layout. From the left, it shows the ranks for the penalty, the ranks for the average edge length, and the ranks for the number of edge crossings, respectively. The ranks of the optimal layout for the penalty are omitted because these were always the first place. The quasi-optimal solutions give considerably good results if we regard only the ranking for the penalty. However, some ranks of quasi-optimal layouts are rather low in regard to the average edge length or number of edge crossings. All cases except the number of edge crossings

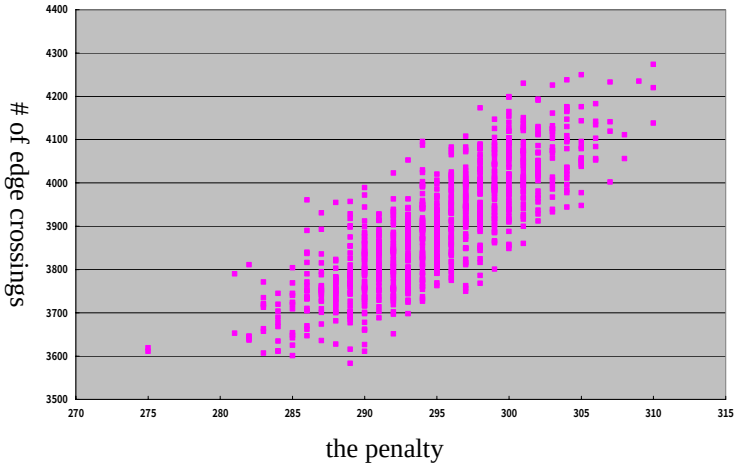


Figure 4: Correlation between the penalty and the number of edge crossings

in G2 are ranked within the 65th place in 1000; the rank of the number of edge crossings in G2 is 144th.

These data led us to conclude that the penalty works moderately well. Moreover, we think that the proposed algorithm works well enough. We guess that the goodness (or badness) of the index caused the ranks concerning the aesthetic criteria to be a little low. Therefore, it will be necessary for us to look for a better alternative index.

By the way, the penalty used in the experiment assumed that $q = 1$ in Expression (3). We also examined a penalty of $q = 2$ and $q = 3$. When q becomes large, both the correlation between the penalty and the average edge length and the correlation between the penalty and the number of edge crossings tended to become small, against the expected effect of q as described in Section 3.3.3.

We also computed the correlation between the average edge length and the number of edge crossings. For all six graphs, we obtained high correlations of more than 0.94. This means that we can use the common measure to decrease both of the average edge length and the number of edge crossings.

Table 3: Ranks of optimal solutions and quasi-optimal solutions concerning penalty, average edge length, and number of edge crossings

	penalty	Average edge length		Number of edge crossings	
	q.-opt.	opt.	q.-opt.	opt.	q.-opt.
G1	2	1	6	1	33
G2	3	1	53	1	144
G3	3	2	18	1	65
G4	1	3	2	9	4
G5	2	1	21	1	26
G6	1	3	4	2	3

5 Application

Below, we show examples of anchored maps that visualize protein interaction networks.

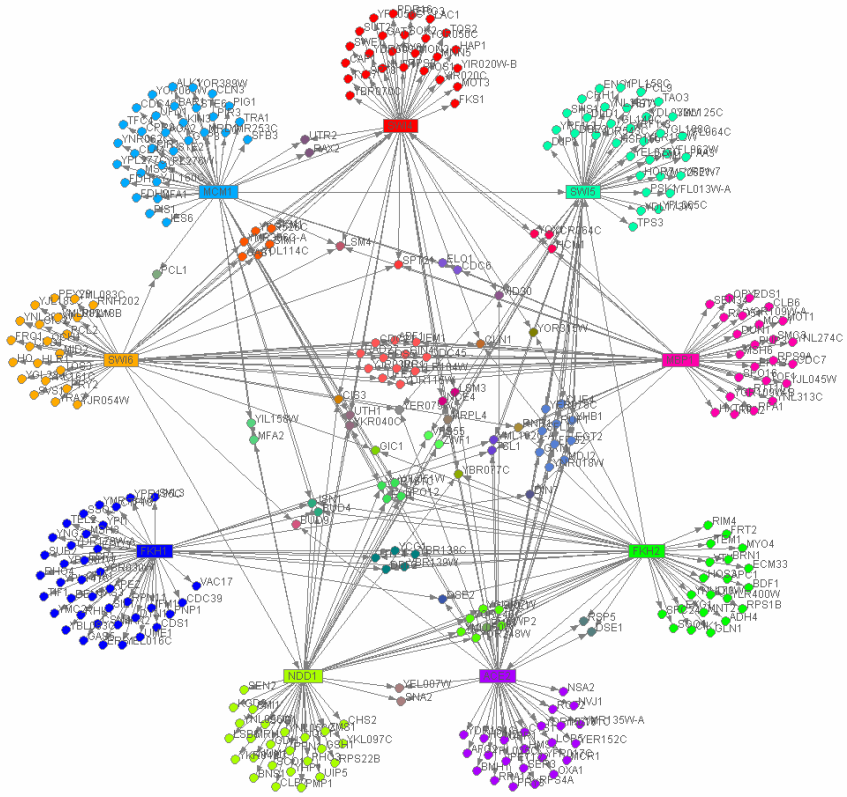
When a protein is generated based on DNA, a transcription factor works. The kind of protein that is generated depends on the transcription factor. A transcription factor generates one or more kinds of protein, and a protein is generated from one or more kinds of transcription factor; there is an n-to-n relation between the proteins and the transcription factors. We guessed that proteins generated from the same transcription factor would have a similar character and that transcription factors generating the same protein would have a similar character. We expected that we could intuitively grasp the global relational structure between proteins and transcription factors if we see such relationships graphically.

We expressed the relationships between the proteins and the transcription factors as anchored maps in which the transcription factors are anchors. Note that, in fact, the relationships between proteins and transcription factors are not bipartite graphs because transcription factors are also proteins.

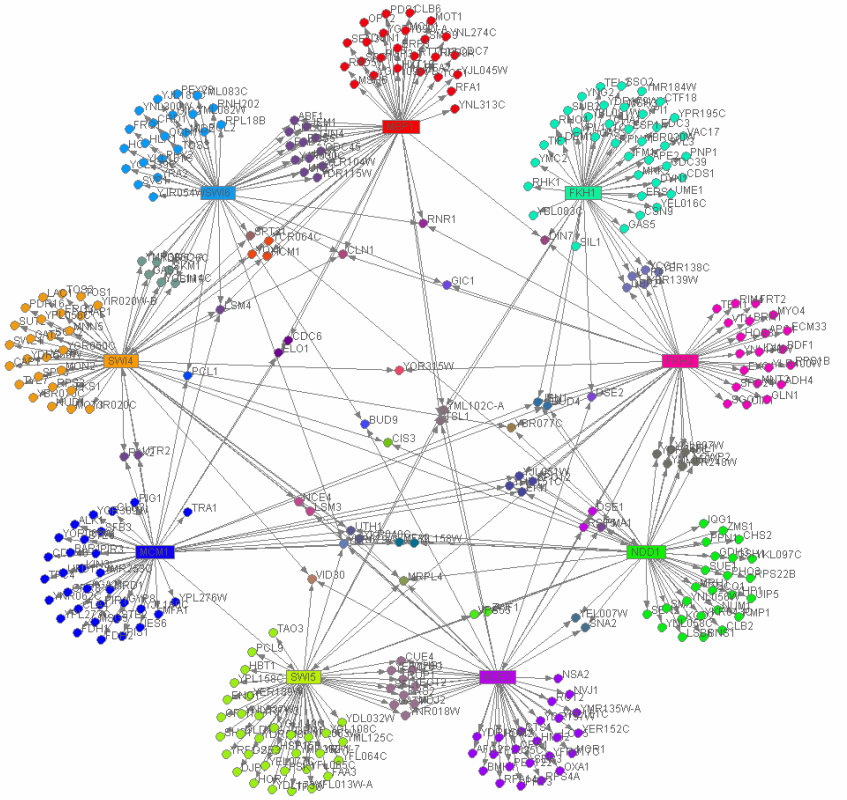
Figure 5 and Figure 6 show interaction networks of yeast proteins. Figure 5 shows the relationships between nine representative transcription factors related to the cell cycle and 276 proteins related to the factors. Figure 6 shows a larger example; it shows 63 transcription factors whose functions are comparatively clear and 1315 proteins related to the factors. As described above, the relations between the proteins and the transcription factors are not bipartite graphs. Here, we deleted the edges toward the transcription factors to make the bipartite graphs. Refer to (Harbison et al 2004) for more information about the protein data.

We drew two kinds of diagrams to see the effect of the proposed technique. In Figure 5(a) and in Figure 6(a), the anchors are arranged at random. In Figure 5(b) and in Figure 6(b), the orders of the anchor nodes are computed by using the proposed technique.

In the random layouts, we can see that the density of the central parts of the diagrams is high, and some transcription factors related to the common proteins are placed far away from each other. As the result, proteins which might have no relationship with each other (in other words, which are generated from quite different transcription factors) gather in a central part of the diagram; we cannot see features like clusters of proteins with similar natures. In contrast, the effect of the proposed technique is obvious. In Figure 5(b) and in Figure 6(b), transcription factors that relate to each other are placed together, and the central parts of the diagrams are sparser. In addition, it seems that the proteins are moderately scattered, and proteins with similar relationships with the transcription factors are placed close to each other.

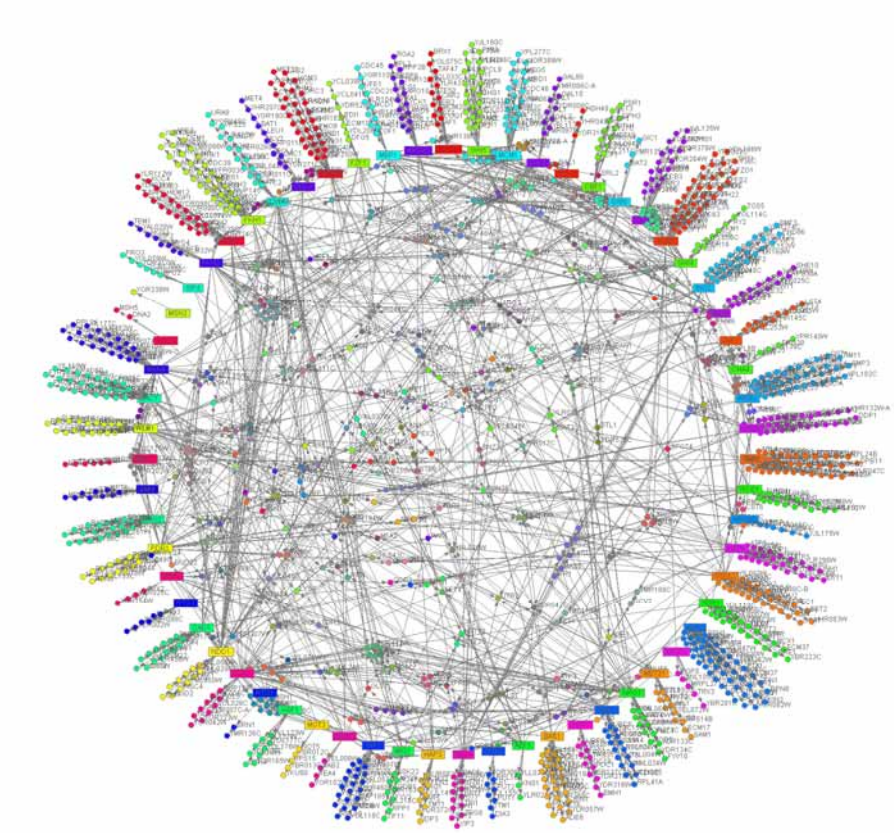


(a) Anchor nodes in random order

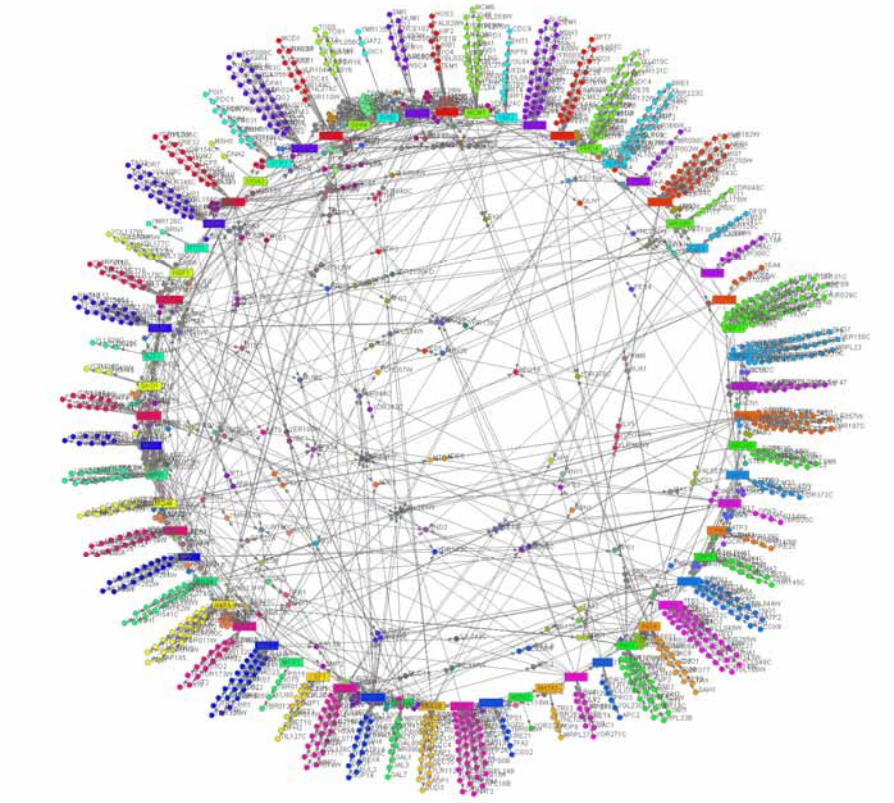


(b) Order of anchor nodes computed by using the proposed technique

Figure 5: Interaction networks of yeast proteins (9 transcription factors)



(a) Anchor nodes in random order



(b) Order of anchor nodes computed by using the proposed technique

Figure 6: Interaction networks of yeast proteins (63 transcription factors)

6 Related work

Visual Who (Donath 1995) is a tool whose purpose is to visualize communities. The tool visually expresses the appearance of the communities based on some mailing lists by using information statistically extracted from the text data of the mailing lists. The positions of the nodes are computed by spring embedding. The user can arrange an arbitrary mailing list as anchors, and the layout of the nodes represent member changes. This tool is interactive, and the user may arrange the anchors manually. It does not provide automatic layout facilities for the anchors.

SQWID (McCreckard & Kehoe 1997) is a Web search tool that expresses the retrieval result of WWW by using anchored maps. Terms used in the query are placed on the vertices of a triangle as anchor nodes. Web pages (or sites) are placed according to a related level with these terms. They limited anchors to three so that the relationships between the arranged Web pages and the fixed terms should not become vague. However, we think that there are a lot of situations in which four or more anchors are needed, and that a automatic technique to find the arrangement of anchor nodes should be developed.

7 Conclusions

We proposed an anchored map drawing technique for bipartite graphs, and explained the two main ideas of the technique. One idea concerns the index to decide the order of anchor nodes. As the index, we defined a penalty based on gaps between anchor nodes connected with a common free node. The other idea is an efficient algorithm to decide the order of anchor nodes by using the index. We implemented these ideas in Java and evaluated the effectiveness of the ideas. We generated six kinds of random bipartite graphs as data for the evaluation experiment. We computed layouts of these graphs by using our technique and one using a random ordering of anchor nodes, and considered the results according to aesthetic criteria. The penalty seems to be useful as an alternative index as long as we can see the correlations between it and the average edge length and between it and the number of edge crossings. Moreover, we think that the proposed algorithm is good enough.

We have yet to complete our evaluations for all aesthetic criteria. In addition, we should try to find an alternative index that is better than the penalty described in this article, and to develop other algorithms to find good orders of anchor nodes based on the index.

Acknowledgements

The author would like to thank Professor Yasubumi Sakakibara and Mr. Yuji Kawada of Keio University for their valuable advice on the interaction network of proteins. This research was supported by the Artificial Intelligence Research Promotion Foundation.

References

Eades, P. (1984), A Heuristic for Graph Drawing, Congressus Numerantium 42, pp. 149-160.

Harbison, C. T. et al. (2004), Transcriptional regulatory code of a eukaryotic gnome, Nature, 431, pp. 99-104.

Misue, K. & Watanabe, I. (1999), Visualization of Keyword Association for Text Mining, In *IPSJ SIG Technical Report 1999-FI-55-8*, Information Processing Society of Japan, pp. 65-72 (in Japanese).

McCrickard, S. D. & Kehoe, C. (1997), Visualizing Search Results using SQWID, In *Proc. of the sixth International World Wide Web Conference (WWW 6)*.

Donath, J. S. (1995), Visual Who: Animating the affinities and activities of an electronic community, ACM Multimedia 95 – Electronic Proceedings.