A High-Dimensional Approach to Interactive Graph Visualization

Hiroshi Hosobe
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
hosobe@nii.ac.jp

ABSTRACT

Graph layout is an information visualization technology for illustrating relations between objects. Interactive graph layout is often important since it is difficult to statically lay out complex graphs such as general undirected graphs. In this paper, we propose a novel approach to interactive layout of general undirected graphs. The basic idea of our approach is to use static graph layouts in high-dimensional spaces to dynamically find two-dimensional layouts according to user interaction. The resulting method that we present exhibits the following two characteristics: (1) it efficiently updates two-dimensional graph layouts during user interaction; (2) it follows users' node dragging operations by actively moving other closely related nodes. Our method adopts eigenvector-based multidimensional scaling to compute high-dimensional graph layouts, and performs constraint satisfaction to determine appropriate two-dimensional planes onto which the high-dimensional layouts will be projected.

Keywords

information visualization, interactive graph layout, general undirected graphs, multidimensional scaling

1. INTRODUCTION

Information visualization is often needed to illustrate relations between objects. *Graphs* are formal means for expressing such relations; they represent objects as nodes and such relations as edges. To visualize information expressed as graphs, researchers have studied *graph layout* or graph drawing technology [1], which automatically computes appropriate positions of nodes and edges. Graph layout methods are designed according to classes of graphs that are determined by their structures. Examples of classes are trees, directed graphs, planar graphs, and *general undirected graphs*.

General undirected graphs, whose edges have no directions, are used to express various information with network structures. To lay out them, the *force-directed approach* [1], which regards graphs as dynamic systems and finds their stable layouts by running simulations, is often adapted, and has been extensively studied. General undirected graph layout methods including the force-directed approach have been successful to a certain degree; they obtain aesthetic layouts of small graphs with tens of nodes and also of mesh-structured graphs with millions of nodes.

Nevertheless, layout of general undirected graphs of certain size and complexity is still a hard problem. Visualizing the structure of such a graph with a single static layout is considered to be difficult because of its high generality.

An effective means for this problem is *interactive graph layout* (also known as dynamic graph layout), which allows users to visualize graphs interactively. The force-directed approach is easily extensible to interactive graph layout by enabling users to affect dynamic simulations, and therefore has been used for this purpose [11].

In this paper, we propose a novel approach to interactive layout of general undirected graphs, which is different from the force-directed one. The basic idea of our approach is to statically compute graph layouts in *high-dimensional spaces* beforehand, and then to dynamically determine two-dimensional layouts according to users' node dragging operations as well as to the high-dimensional layouts.

The resulting method that we present exhibits the following two characteristics.

1. It efficiently computes two-dimensional graph layouts, and processes graphs with more than one thousand nodes within a few tens of milliseconds. Therefore, it handles user interaction in real time.

2. It follows users' node dragging operations by actively moving other closely related nodes. This property is applicable, for example, to emphasize a part of a graph by dragging only a few nodes of it.

Our method computes internal high-dimensional graph layouts by adopting a graph layout method [9] based on *multidimensional scaling* [12] using *eigenvector* calculation and by extending it to high dimensions. Also, it transforms such high-dimensional layouts into two-dimensional ones by *projecting* them onto appropriate two-dimensional planes that we determine by constraint satisfaction.

The rest of this paper is organized as follows. Section 2 describes related work on graph layout. Section 3 explains an existing graph layout method on which our research is based. Section 4 proposes our interactive graph layout method, and Section 5 evaluates it by presenting experimental results.

SAC'04, March 14-17, 2004, Nicosia, Cyprus

Section 6 discusses our method. Finally, Section 7 mentions the conclusions and future work of this research.

2. RELATED WORK

The force-directed approach [1] is often adopted to find layouts of general undirected graphs. Eades proposed a method, known as the spring embedder, that finds a stable layout of nodes by using attractive and repulsive forces of springs assigned to edges [2]. Kamada and Kawai presented a method that uses springs to make the Euclidean distances between any pairs of nodes close to the graph-theoretic distances [7]. In [4], a method is provided that first finds graph layouts in multidimensional (e.g., four-dimensional) spaces by using the force-directed approach and then projects the layouts onto two- or three-dimensional spaces. An example of applying the force-directed approach to interactive information visualization is DocSpace [11], which visualizes relations between documents such as scientific papers.

Multidimensional scaling (MDS) [12] is sometimes used for information visualization including graph layout. MDS is a statistical method that finds multidimensional layouts of objects whose similarities are given as input data. Kruskal and Seery proposed adopting MDS for graph layout, and gave an actual method that uses MDS based on eigenvector calculation [9] (which Section 3 describes in detail). An example that uses MDS for information visualization is SemNet [3], which visualizes large knowledge bases by adopting graph layouts in three dimensional spaces.

Methods using eigenvectors for graph layout are attracting attention. In [10], an example is presented that finds a graph layout in the football shape by adopting eigenvectors of Laplacian matrices. The ACE algorithm [8], which is based on a similar formulation, computes layouts of graphs with more than 10^6 nodes within a minute by using an algebraic multigrid algorithm to speed up eigenvector calculation. In [5], a method is given that finds layouts of graphs with 10^5 nodes within a few seconds by first computing graph layouts of relatively high dimensions such as 50 and then by projecting them onto two-dimensional planes according to principal component analysis using eigenvector calculation. It should be noted, however, that these results were obtained by applying the methods to special graphs with mesh structures.

Fisheyeing is known as an effective means for visualizing complex information. It enables users to expand and emphasize details of visualized information and, at the same time, to display its whole structure by simplifying the other part. It has also been applied to graph visualization [3, 6].

3. MULTIDIMENSIONAL GRAPH LAYOUT

This section explains an existing multidimensional graph layout method that we use in this research.

3.1 Torgerson's Method

As the basis of multidimensional graph layout, we describe Torgerson's method [9, 12], also known as metric multidimensional scaling and as principal coordinate analysis in the field of statistics. Given distances between any pairs of objects, it finds a layout of them that satisfies the distances.

Assume that we have distances d_{ij} between any pairs i and j of n objects, and also that they satisfy the distance axioms. First, define a_{ij} as follows:

$$a_{ij} = \frac{1}{2}\left(\frac{1}{n}\sum_k d_{ik}^2 + \frac{1}{n}\sum_k d_{kj}^2 - \frac{1}{n^2}\sum_k\sum_l d_{kl}^2 - d_{ij}^2\right).$$

Next, define an $n \times n$ real symmetric matrix $A = (a_{ij})$. Then A is diagonalizable as $X^T A X = \Lambda$ for an orthogonal matrix X, where, with the eigenvalues λ_k of A and the eigenvectors $\boldsymbol{x}_k$ corresponding to λ_k, Λ is the diagonal matrix with λ_k as its (k,k) elements, and $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$.

Now, let $P = X\Lambda^{1/2}$, where $\Lambda^{1/2}$ is the diagonal matrix with $\sqrt{\lambda_k}$ as its (k,k) elements. Then, for an ideal set of d_{ij}, each eigenvalue λ_k is nonnegative, and each i-th row $(p_{i1}, p_{i2}, \ldots, p_{in})$ of P can be regarded as the coordinates of the location of the object i in the n-dimensional real Euclidean space. It should be noted that actual data usually result in occurrences of negative eigenvalues.

Ordinary applications use only the coordinates corresponding to a small number of the largest eigenvalues. For example, the first and second largest eigenvalues λ_1 and λ_2 obtain a two-dimensional layout with the i-th objects located at (p_{i1}, p_{i2}).

3.2 Graph Layout Using Torgerson's Method

Kruskal and Seery proposed a method that uses Torgerson's method to lay out connected general undirected graphs [9] (the TKS method). It is realized as follows.

1. Given a graph, first compute the graph-theoretic distances (the lengths of the shortest paths) between any pairs of its nodes.
2. Next, perform Torgerson's method by using the graph-theoretic distances, to obtain a layout of the nodes on a two-dimensional plane.

Although they assumed two dimensions, the method is easily extensible to multidimensional graph layouts.

3.3 Relation to and Comparison with Kamada and Kawai's Method

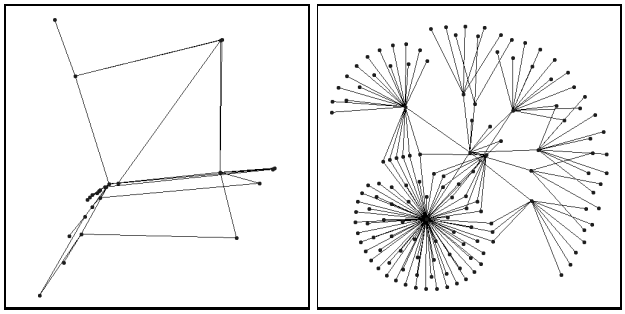
The TKS method has close relation to Kamada and Kawai's method [7] (the KK method) described in Section 2; both of them attempt to find layouts of nodes in such a way that the Euclidean distances between the nodes should be the graph-theoretic ones. The principal difference is that the former adopts spaces of necessary dimensions whereas the latter uses only two-dimensional planes.

The TKS method is considered to obtain general outlines of graphs on two-dimensional planes. More local details of the graphs are represented in the higher dimensions, and therefore are invisible from the two-dimensional layouts.

By contrast, the KK method well represents local details of graphs in two-dimensional layouts. However, it sometimes breaks their general outlines, giving near locations to nodes that are distant in the graph-theoretic sense. The causes are considered that it weakens springs between nodes separated by long graph-theoretic distances, and also that it often finds local optimal solutions in computing the layouts.

For a comparison between the TKS and KK methods, Figure 1 presents examples of two-dimensional layouts obtained by applying these methods to the same graph. The graph treated here is the AT&T graph[1] `ug_263`, a general undirected graph with 141 nodes and 177 edges.

[1] `ftp://ftp.research.att.com/dist/drawdag/ug.gz`



(a) (b)

Figure 1: Layouts of the AT&T graph ug_263 (with 141 nodes and 177 edges) obtained by the (a) TKS and (b) KK methods.

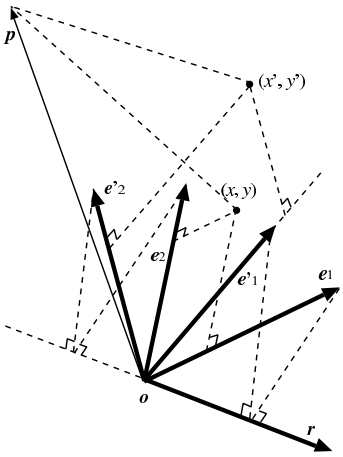


Figure 2: Updating the projection plane.

4. OUR METHOD

This section proposes a high-dimensional approach to interactive visualization of graphs on two-dimensional planes.

4.1 Computing Two-Dimensional Layouts

Our new method computes two-dimensional graph layouts by projecting graph layouts in high-dimensional spaces onto two-dimensional planes. It handles connected general undirected graphs, and represents edges as straight lines connecting nodes.

Adopting the TKS method described in the previous section, our method computes graph layouts in high-dimensional spaces. Characteristically, it uses all the coordinates corresponding to positive eigenvalues. Generally, since the TKS method exploits graph-theoretic distances in Torgerson's method, it obtains many positive eigenvalues, which means that the dimensionalities of the resulting graph layouts are high. In the following, we assume that the eigenvalues $\lambda_1, \lambda_2, \ldots$ are sorted in descending order, and also $d \geq 2$, where d is the number of positive eigenvalues. Then we have $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d > 0$, and the position of each node i in the high-dimensional space is $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id})$.

We project such a d-dimensional graph layout onto a two-dimensional plane (that we call the projection plane) as follows. Consider the projection plane as the plane spanned by two d-dimensional vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. Using these vectors, we can obtain the two-dimensional coordinates of node i as $(\boldsymbol{p}_i \cdot \boldsymbol{e}_1, \boldsymbol{p}_i \cdot \boldsymbol{e}_2)$.

To initialize $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ for the initial two-dimensional layout, we let $\boldsymbol{e}_1 = \boldsymbol{f}_1/|\boldsymbol{f}_1|$ and $\boldsymbol{e}_2 = \boldsymbol{f}_2/|\boldsymbol{f}_2|$, with $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ which are the d-dimensional vectors defined as $\boldsymbol{f}_1 = (\lambda_1^\alpha, 0, \lambda_3^\alpha, 0, \ldots)$ and $\boldsymbol{f}_2 = (0, \lambda_2^\alpha, 0, \lambda_4^\alpha, \ldots)$, where α is a parameter to adjust how the coordinates affect the two-dimensional layout. Note that $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are orthogonal.

In general, the above definition follows that, with a larger parameter α, high-dimensional coordinates less affect the initial two-dimensional graph layout. Especially, when $\lambda_1 > \lambda_3$ and $\lambda_2 > \lambda_4$, the limit of the initial layout as $\alpha \to \infty$ is the two-dimensional layout obtained by the TKS method. We use $\alpha = 1/2$ by default since computing appropriate α is generally difficult.

4.2 Updating Two-Dimensional Layouts

Next, we present a method that enables users to interactively update two-dimensional graph layouts. It is realized by moving projection planes. Since it is not necessary to modify graph layouts in high-dimensional spaces, our method provides high efficiency in updating two-dimensional layouts. It allows a user to drag a single node at a time.

The basic idea of our method is that it rotates the projection plane in the three-dimensional space spanned by the current vectors for the projection plane and the vector positioning the dragged node. To compute this, it performs constraint satisfaction by imposing the constrains that should be satisfied by the vectors spanning the projection plane.

We describe it in detail below. First, we define constants that work as input. Let $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ be the current vectors spanning the projection plane. Let $\boldsymbol{p}$ be the position of the dragged node in the d-dimensional space, and (x, y) and (x', y') be its current and new two-dimensional coordinates respectively. Then we have $(x, y) = (\boldsymbol{p} \cdot \boldsymbol{e}_1, \boldsymbol{p} \cdot \boldsymbol{e}_2)$ by definition. Also, we assume $\|(x, y)\| < \|\boldsymbol{p}\|$ and $\|(x', y')\| < \|\boldsymbol{p}\|$.

Next, let $\boldsymbol{e}'_1$ and $\boldsymbol{e}'_2$ be the new vectors spanning the projection plane. We consider these vectors to be in the three-dimensional space spanned by $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, and $\boldsymbol{p}$. Then they can be expressed with six variables a_1, b_1, c_1, a_2, b_2, and c_2 as $\boldsymbol{e}'_1 = a_1\boldsymbol{e}_1 + b_1\boldsymbol{e}_2 + c_1\boldsymbol{p}$ and $\boldsymbol{e}'_2 = a_2\boldsymbol{e}_1 + b_2\boldsymbol{e}_2 + c_2\boldsymbol{p}$. Also, let $\boldsymbol{r}$ be the vector indicating the rotation axis of the projection plane. Then it can be represented with two variables s and t as $\boldsymbol{r} = s\boldsymbol{e}_1 + t\boldsymbol{e}_2$.

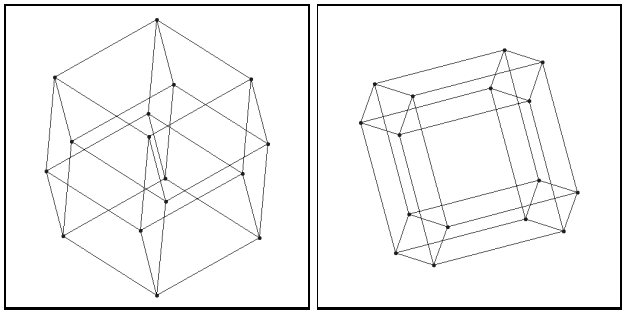
Now, using these constants and variables, we impose the following eight constraints:

$$\begin{array}{lll} \|\boldsymbol{e}'_1\| = 1, & \|\boldsymbol{e}'_2\| = 1, & \boldsymbol{e}'_1 \cdot \boldsymbol{e}'_2 = 0, \\ \|\boldsymbol{r}\| = 1, & \boldsymbol{e}'_1 \cdot \boldsymbol{r} = \boldsymbol{e}_1 \cdot \boldsymbol{r}, & \boldsymbol{e}'_2 \cdot \boldsymbol{r} = \boldsymbol{e}_2 \cdot \boldsymbol{r}, \\ \boldsymbol{p} \cdot \boldsymbol{e}'_1 = x', & \boldsymbol{p} \cdot \boldsymbol{e}'_2 = y'. & \end{array}$$

The first three constraints mean that $\boldsymbol{e}'_1$ and $\boldsymbol{e}'_2$ are unit vectors and orthogonal. The next three constraints indicate that $\boldsymbol{r}$ is a unit vector, and that $\boldsymbol{e}'_1$ and $\boldsymbol{e}'_2$ are the rotations of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ around $\boldsymbol{r}$. The last two constraints imply that (x', y') is the coordinates obtained by projecting $\boldsymbol{p}$ onto the new projection plane. These vectors are depicted in Figure 2 in the three-dimensional manner.

We can solve these eight constraints easily. We have only eight variables a_1, b_1, c_1, a_2, b_2, c_2, s, and t, and only equality constraints, the first six of which are quadratic, and the last two of which are linear. Therefore, we can efficiently solve them by a basic numerical method for simultaneous nonlinear equations such as Newton's method.

In the above, we assumed two conditions $\|(x, y)\| < \|\boldsymbol{p}\|$



(a) (b)

Figure 3: Interactive layout of the graph with the four-dimensional hypercubic structure.

and $\|(x', y')\| < \|\boldsymbol{p}\|$. The condition $\|(x, y)\| < \|\boldsymbol{p}\|$ is used to guarantee that $\boldsymbol{p}$ is linearly independent of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, and is needed to enable the projection plane to rotate in the three-dimensional manner. This condition does not hold if and only if $\|(x, y)\| = \|\boldsymbol{p}\|$. Since it rarely occurs, we can guarantee the condition by prohibiting the dragging of such a node. The other condition $\|(x', y')\| < \|\boldsymbol{p}\|$ is used to make certain of the existence of the new projection plane, and also to allow dragging the same node repeatedly. To guarantee this condition, we only need to introduce a predetermined small positive constant τ (e.g., $\tau = 10^{-3}$), and then to replace (x', y') with $\{(1-\tau)\|\boldsymbol{p}\|/\|(x', y')\|\}(x', y')$ in the case that $\|(x', y')\| > (1-\tau)\|\boldsymbol{p}\|$ holds.

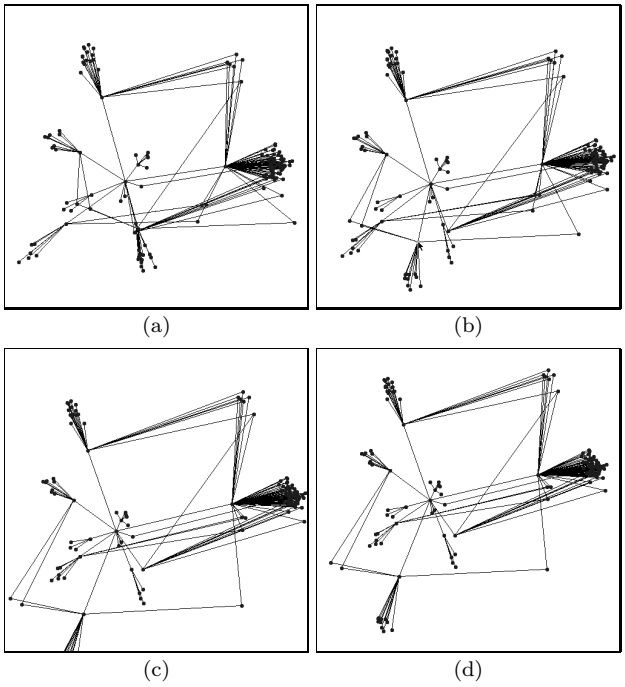
5. EXPERIMENTAL EVALUATION

To evaluate the interactive graph layout method proposed in the previous section, we implemented a prototype program in C++, and performed experiments. It adopts ATLAS 3.4.1 and LAPACK 3.0 for linear computation including eigenvector calculation, uses single precision for floating point arithmetic, and exploits SIMD instructions (SSE) inside ATLAS. Also, it employs Floyd's algorithm to obtain graph-theoretic distances. We compiled the program by using GCC 2.95.3 with the –O3 option, and executed it on a 1.13 GHz Pentium III-M processor running Linux 2.2.25.

Below we provide three examples of executing our method. In these examples, we use $\alpha = 1/2$ as the parameter to obtain the initial two-dimensional graph layouts, which we mentioned in the previous section.

The first example is to lay out the graph whose structure is analogous to the four-dimensional hypercube. The graph has 16 nodes and 32 edges. The initial graph layout generated by our method is shown in Figure 3(a). Next, the layout obtained by dragging a node is given in Figure 3(b). By updating the layout as illustrated in Figure 3(b), it is facilitated to view the structure of the graph as the one that connects four rectangles by the corresponding vertices, or as the one that connects two three-dimensional boxes similarly. The times needed to compute the initial and updated graph layouts were both less than 10 milliseconds.

The next example is to lay out the AT&T graph `ug_263`, which we also used in Section 3. Its initial layout is presented in Figure 4(a). The internal dimensionality of the graph (i.e., d in the previous section) is 125. This two-dimensional graph layout exposes local details of the graph that the TKS method hides in high-dimensional coordinates,



(a) (b)

(c) (d)

Figure 4: Interactive layout of the AT&T graph `ug_263` (with 141 nodes and 177 edges).

as suggested by the comparison with the layout in Figure 1(a). The following figures show interactive layout of the graph by dragging a single node. Figure 4(b) illustrates the middle of dragging the node, and Figure 4(c) gives the layout right after the dragging, which was then scaled to the screen as depicted in Figure 4(d). This example shows that our method updated the two-dimensional layout by actively moving nodes related closely to the dragged one. The times required for these operations are presented in Table 1.

The final example is the layout of the AT&T graph `ug_380`, which consists of 1,104 nodes and 3,231 edges. Figure 5(a) shows the initial layout of the graph, whose internal dimensionality is 697. Figure 5(b) gives a graph layout obtained by dragging a single node on the lower right side and then by scaling the resulting layout to the screen. Also, Figures 5(c) and (d) illustrate layouts after dragging other two nodes one by one. This example indicates the use of our method to expand and emphasize a part of a graph. The times needed for these processes are provided in Table 1. The result shows that our method efficiently performed updating the layout of the graph even in the case that much time was needed to compute the initial layout of the graph.

6. DISCUSSION

In our method, high dimensionalities of internal graph layouts are important for facilitating interactive graph visualization. This is because, in general, higher dimensionalities result in greater freedom to locate nodes on two-dimensional planes. It may be understood by the fact that, if the internal dimensionality is equal to the number of nodes and the vectors indicating the node positions are linearly independent of each other, we can obtain any two-dimensional layouts by determining appropriate projection planes.

It is possible to obtain high-dimensional graph layouts by

Table 1: Running times of our interactive graph layout method.

Graphs	Numbers of nodes	Numbers of edges	Internal dimensionalities	Times for initial layouts (for graph-theoretic distances and eigenvectors)	Times for updating layouts
ug_263	141	177	125	50 ms (20 ms, 30 ms)	$\leq$ 10 ms
ug_380	1,104	3,231	697	29.7 s (8.6 s, 21.0 s)	$\leq$ 20 ms

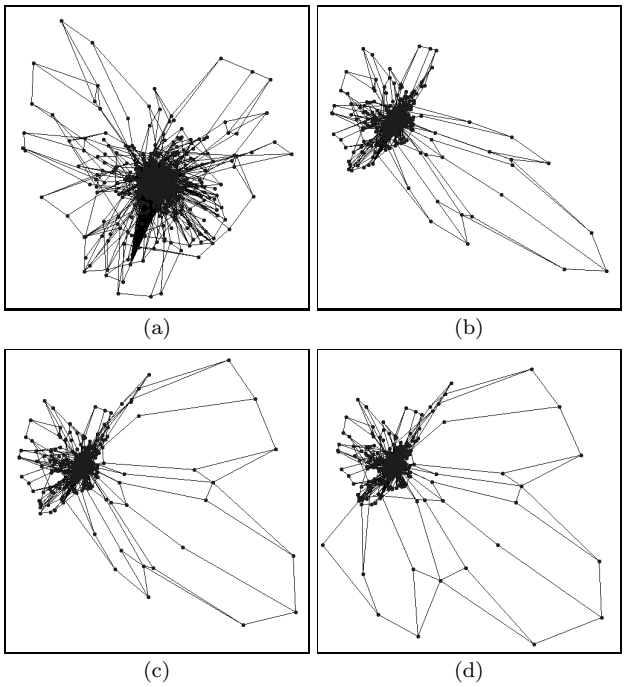


Figure 5: Interactive layout of the AT&T graph ug_380 (with 1,104 nodes and 3,231 edges).

other methods than the TKS. However, the TKS method is currently the only method that we know to fit our approach. In other words, when another high-dimensional method is used, it will be possible that transforming two-dimensional layouts by moving projection planes will not work appropriately.[2] Nevertheless, since speeding up the computation of high-dimensional graph layouts and also improving the quality of initial two-dimensional layouts are needed to further improve the usefulness of our approach, it is important to pursue other high-dimensional graph layout methods that work well for our approach.

Although we handled general undirected graphs in this research, our method can be accommodated to other kinds of data that are applicable to Torgerson's method; it can handle data that express similarities of objects. It should be noted, however, that Torgerson's method assumes similarity data with an analogous nature to that of Euclidean distances (graph-theoretic distances are considered to conform to this assumption). Also, it should be remembered that edges play an important role in visualizing relations between nodes in graph layouts. Therefore, even when we handle other kinds of data, we will need to display some alternative to edges that represents close similarities between objects.

[2]We found that the method using eigenvectors of Laplacian matrices, presented in Section 2, does not work well for our high-dimensional approach even if we apply a similar idea to it.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a high-dimensional approach to interactive layout of general undirected graphs. The resulting method efficiently updates two-dimensional graph layouts, and also transforms them appropriately according to user interaction.

A future direction of this research is to speed up computing high-dimensional graph layouts. To do it, we examine if existing methods other than the TKS are appropriate to generating high-dimensional layouts for our purpose. Our plan also includes extending our prototype program by enhancing its display and user interaction functions.

8. REFERENCES

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

[2] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[3] K. M. Fairchild, S. E. Poltrock, and G. W. Furnas. SemNet: Three-dimensional graphic representation of large knowledge bases. In *Cognitive Science and Its Applications for Human-Computer Interaction*, pages 201–233. Lawrence Erlbaum, 1988.

[4] P. Gajer, M. T. Goodrich, and S. G. Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. In *Graph Drawing—GD2000*, volume 1984 of *LNCS*, pages 211–221. Springer, 2000.

[5] D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. In *Graph Drawing—GD2002*, volume 2528 of *LNCS*, pages 207–219. Springer, 2002.

[6] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Trans. Visual. Comput. Gr.*, 6(1):24–43, 2000.

[7] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989.

[8] Y. Koren, L. Carmel, and D. Harel. ACE: A fast multiscale eigenvectors computation for drawing huge graphs. In *Proc. IEEE InfoVis*, pages 137–144, 2002.

[9] J. B. Kruskal and J. B. Seery. Designing network diagrams. In *Proc. 1st General Conf. on Social Graphics*, pages 22–50. U.S. Dept. of the Census, 1980.

[10] T. Pisanski and J. Shawe-Taylor. Characterizing graph drawing with eigenvectors. *J. Chem. Inf. Comput. Sci.*, 40(3):567–571, 2000.

[11] J. Tatemura. Visualizing document space by force-directed dynamic layout. In *Proc. IEEE VL*, pages 119–120, 1997.

[12] F. W. Young. Multidimensional scaling. In *Encyclopedia of Statistical Sciences*, volume 5, pages 649–658. Wiley, 1985.