An Extended High-Dimensional Method for Interactive Graph Drawing

Hiroshi Hosobe

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

Abstract

Graph drawing is an information visualization technology for illustrating relations between objects. Interactive graph drawing is often important since it is difficult to statically lay out complex graphs. For the interactive drawing of general undirected graphs, we previously proposed the high-dimensional approach, which used static graph layouts in high-dimensional spaces to dynamically find two-dimensional layouts according to user interaction. Although the resulting interactive graph drawing method was fast, it imposed a limitation on the successive manipulation of dense parts of graphs. In this paper, we propose an extended method to tackle the limitation. Our new method enables multiple graph nodes to be controlled simultaneously in interactive graph drawing. For this purpose, we extend the underlying constraint programming mechanism by introducing the notion of soft constraints as well as by using additional constraints on multiple controlled nodes.

Keywords: interactive graph drawing, general undirected graphs, high-dimensional approach

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 drawing* (Di Battista, Eades, Tamassia & Tollis 1999), which automatically computes appropriate positions of nodes and edges. Graph drawing 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. Although previous methods including the force-directed approach (Di Battista et al. 1999) have been successful to a certain degree, drawing complex general undirected graphs with more than hundreds of nodes is still a hard problem; visualizing the structure of such a graph with a single static layout is difficult because of its high generality. An effective means for this problem is *interactive graph drawing*, which allows users to visualize graphs interactively.

For this purpose, we previously proposed the *high-dimensional approach* (Hosobe 2004), which used static graph layouts in high-dimensional spaces to dynamically find two-dimensional layouts according to user interaction. The resulting interactive graph drawing method exhibited the following two properties: first, it efficiently updates two-dimensional graph layouts, and processes graphs with more than one thousand nodes within a few tens of milliseconds; second, it follows users' node dragging operations by actively moving other closely related nodes. To transform such high-dimensional layouts into two-dimensional ones, it projects them onto appropriate two-dimensional planes that it determines by constraint satisfaction.

However, our previous high-dimensional method imposed a limitation on the successive manipulation of dense parts of graphs; while a user is dragging a node in a dense part, the method sometimes largely moves other nodes that the user has dragged before. This is a side effect caused by the property that the method tends to move nodes closely related to dragged ones.

In this paper, we propose an extended high-dimensional method to tackle that limitation. Our new method enables multiple graph nodes to be controlled simultaneously in interactive graph drawing. For this purpose, we modify the underlying constraint programming mechanism by applying the following two extensions:

- Using additional constraints on multiple controlled nodes;
- Introducing the notion of *soft constraints* (Barták 2002).

We demonstrate the usefulness of our extended method by presenting experimental results.

The rest of this paper is organized as follows. Section 2 describes related work on graph layout. Section 3 explains the basic high-dimensional method that we proposed previously. Section 4 proposes our extended high-dimensional method. Section 5 provides its implementation, and Section 6 presents experimental results. Section 7 discusses our extended method. Finally, Section 8 mentions the conclusions and future work of this research.

2 Related Work

The force-directed approach (Di Battista et al. 1999) 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 (Eades 1984). 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 (Kamada & Kawai 1989).

The force-directed approach is applicable to drawing graphs of three or higher dimensions. GEM-3D (Bruß & Frick 1996) uses a randomized adaptive spring-embedder algorithm to obtain three-dimensional graph layouts. In (Gajer, Goodrich & Kobourov 2000), 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.

In (Harel & Koren 2002), 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.

3 The Basic High-Dimensional Method

This section explains the basic high-dimensional method for interactive graph drawing (Hosobe 2004).

3.1 Multidimensional Graph Layout

The basic high-dimensional method uses Torgerson's method (Kruskal & Seery 1980, Young 1985) as its fundamental basis. Torgerson's method is 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.

We describe Torgerson's method below. 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:

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

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 Λ is a diagonal matrix. With the eigenvalues λ_k of A and the eigenvectors $\boldsymbol{x}_k$ corresponding to λ_k, such X and Λ are obtained as follows: Λ has λ_k as its (k,k) elements, and $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$.

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. Torgerson's method regards each i-th row $(p_{i1}, p_{i2}, \ldots, p_{in})$ of P 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 based on Torgerson's method 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}).

Kruskal and Seery proposed a method that uses Torgerson's method to lay out connected general undirected graphs (Kruskal & Seery 1980) (called the TKS method). It is realized as follows.

1. Given a graph, first compute the graph-theoretic distances (or 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.

The time complexity of this multidimensional graph layout method is typically $O(n^3)$. More specifically, the time complexity of computing graph-theoretic distances is $O(n^3)$ if Floyd's algorithm is used (Ishihata 1989). The time complexity of obtaining the matrix A is $O(n^2)$. Strictly, the time needed to compute the eigenvalues and eigenvectors depends on actual A; however, if, e.g., the Jacobi method is exploited, it is typically performed with $O(n^3)$ operations (Press, Teukolsky, Vetterling & Flannery 1992).

3.2 Interactive Graph Drawing

The basic high-dimensional method (Hosobe 2004) 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 subsection, the basic high-dimensional method computes graph layouts in high-dimensional spaces. 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. Assume that the eigenvalues λ_1, λ_2, ... are sorted in descending order, and also $d \geq 2$, where d is the number of positive eigenvalues; that is, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d > 0$. Then the position of each node i in the high-dimensional space is $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id})$.

The method projects such a d-dimensional graph layout onto a two-dimensional plane (called the projection plane) as follows. Consider the projection plane as the plane spanned by two orthonormal d-dimensional vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. Using these vectors, the two-dimensional coordinates of node i are obtained as $(\boldsymbol{p}_i \cdot \boldsymbol{e}_1, \boldsymbol{p}_i \cdot \boldsymbol{e}_2)$.

For the initial two-dimensional layout, $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are initialized by letting $\boldsymbol{e}_1 = \boldsymbol{f}_1/\|\boldsymbol{f}_1\|$ 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 $\boldsymbol{f}_2 = (0, \lambda_2^\alpha, 0, \lambda_4^\alpha, \ldots)$. Here α is a parameter, typically set to $1/2$, to adjust how the coordinates affect the two-dimensional layout. Note that $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are orthonormal.

To enable users to interactively update two-dimensional graph layouts, the method moves projection planes. Since it is not necessary to modify graph layouts in high-dimensional spaces, it provides high efficiency in updating two-dimensional layouts. It allows a user to drag a single node at a time. The basic idea 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 constraints that should be satisfied by the vectors spanning the projection plane.

4 Constraining Multiple Nodes

In this section, we extend the basic high-dimensional method explained in the previous section. Basically, we extend our method in such a way that it can constrain multiple nodes at a time. For this purpose, the

extended method rotates the projection plane in a higher-dimensional space that are constructed by using these multiple constrained nodes. However, only incorporating additional constraints is not sufficient for our extension. This is because it adds complexity to the solution space, which results in difficulty in dragging nodes. To remedy this problem, we also introduce soft constraints (Barták 2002), which relax the solution space and thus facilitate dragging nodes.

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. Unlike the original high-dimensional method, the new method does not require $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ to be orthonormal; that is, these vectors may have lengths unequal to 1, and also may be non-orthogonal. Then the two-dimensional coordinates (x_i, y_i) of node i (whose high-dimensional position is $\boldsymbol{p}_i$) are obtained by solving

$$\begin{aligned} \|\boldsymbol{e}_1\|^2 x_i + (\boldsymbol{e}_1 \cdot \boldsymbol{e}_2)\, y_i &= \boldsymbol{p}_i \cdot \boldsymbol{e}_1 \\ (\boldsymbol{e}_1 \cdot \boldsymbol{e}_2)\, x_i + \|\boldsymbol{e}_2\|^2 y_i &= \boldsymbol{p}_i \cdot \boldsymbol{e}_2, \end{aligned}$$

which are solvable if $\|\boldsymbol{e}_1\|^2\|\boldsymbol{e}_2\|^2 - \boldsymbol{e}_1 \cdot \boldsymbol{e}_2 \neq 0$.

Let $i_1, i_2, \ldots, i_m$ be the indices of constrained nodes, and (x'_{i_j}, y'_{i_j}) be the new two-dimensional coordinates of the i_j-th node. We assume that $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{p}_{i_1}, \boldsymbol{p}_{i_2}, \ldots, \boldsymbol{p}_{i_m}$ are linearly independent. From these vectors, we obtain a set of orthonormal vectors $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_{m+2}$. It is done by using the Gram-Schmidt algorithm as follows:

$$\begin{aligned} \boldsymbol{\epsilon}_1 &= \frac{\boldsymbol{\epsilon}'_1}{\|\boldsymbol{\epsilon}'_1\|} \quad \text{where } \boldsymbol{\epsilon}'_1 = \boldsymbol{e}_1 \\ \boldsymbol{\epsilon}_2 &= \frac{\boldsymbol{\epsilon}'_2}{\|\boldsymbol{\epsilon}'_2\|} \quad \text{where } \boldsymbol{\epsilon}'_2 = \boldsymbol{e}_2 - (\boldsymbol{e}_2 \cdot \boldsymbol{\epsilon}_1)\, \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_{j+2} &= \frac{\boldsymbol{\epsilon}'_{j+2}}{\|\boldsymbol{\epsilon}'_{j+2}\|} \quad (\text{for } j = 1, 2, \ldots, m) \\ &\text{where } \boldsymbol{\epsilon}'_{j+2} = \boldsymbol{p}_{i_j} - \sum_{k=1}^{j+1} \left(\boldsymbol{p}_{i_j} \cdot \boldsymbol{\epsilon}_k\right) \boldsymbol{\epsilon}_k. \end{aligned}$$

Note that we have $\|\boldsymbol{\epsilon}_i\| = 1$ and $\boldsymbol{\epsilon}_i \cdot \boldsymbol{\epsilon}_j = 0$ for any i and j.

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 $(m+2)$-dimensional space spanned by $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_{m+2}$. Then they can be expressed with $2m+4$ variables $\alpha_1, \alpha_2, \ldots, \alpha_{m+2}, \beta_1, \beta_2, \ldots, \beta_{m+2}$ as

$$\begin{aligned} \boldsymbol{e}'_1 &= \sum_{j=1}^{m+2} \alpha_j \boldsymbol{\epsilon}_j \\ \boldsymbol{e}'_2 &= \sum_{j=1}^{m+2} \beta_j \boldsymbol{\epsilon}_j. \end{aligned}$$

Also, let $\boldsymbol{r}$ be the vector indicating the rotation axis of the projection plane. Then it can be represented with two variables γ_1 and γ_2 as follows:

$$\boldsymbol{r} = \gamma_1 \boldsymbol{\epsilon}_1 + \gamma_2 \boldsymbol{\epsilon}_2.$$

Now, using these constants and variables, we impose the $2m$ hard constraints

$$\|\boldsymbol{e}'_1\|^2 x'_{i_j} + (\boldsymbol{e}'_1 \cdot \boldsymbol{e}'_2)\, y'_{i_j} = \boldsymbol{p}_{i_j} \cdot \boldsymbol{e}'_1 \tag{1}$$

$$(\boldsymbol{e}'_1 \cdot \boldsymbol{e}'_2)\, x'_{i_j} + \|\boldsymbol{e}'_2\|^2 y'_{i_j} = \boldsymbol{p}_{i_j} \cdot \boldsymbol{e}'_2 \tag{2}$$

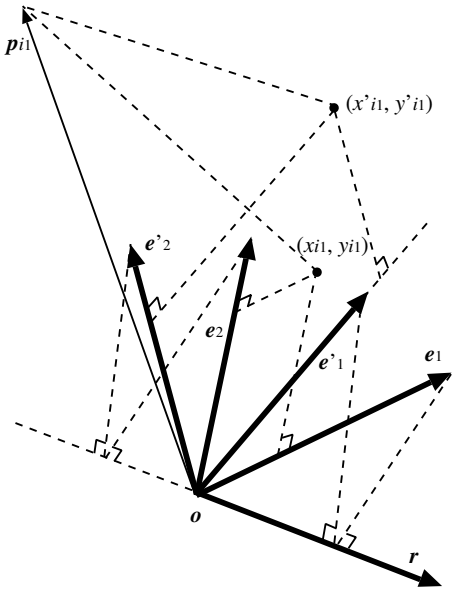


Figure 1: Updating the projection plane.

for $j = 1, 2, \ldots, m$, and also the six soft constraints

$$\|\boldsymbol{e}'_1\| = 1 \tag{3}$$

$$\|\boldsymbol{e}'_2\| = 1 \tag{4}$$

$$\frac{\boldsymbol{e}'_1 \cdot \boldsymbol{e}'_2}{\|\boldsymbol{e}'_1\|\|\boldsymbol{e}'_2\|} = 0 \tag{5}$$

$$\|\boldsymbol{r}\| = 1 \tag{6}$$

$$\frac{\boldsymbol{e}'_1 \cdot \boldsymbol{r}}{\|\boldsymbol{e}'_1\|\,\|\boldsymbol{r}\|} = \frac{\boldsymbol{e}_1 \cdot \boldsymbol{r}}{\|\boldsymbol{e}_1\|\,\|\boldsymbol{r}\|} \tag{7}$$

$$\frac{\boldsymbol{e}'_2 \cdot \boldsymbol{r}}{\|\boldsymbol{e}'_2\|\,\|\boldsymbol{r}\|} = \frac{\boldsymbol{e}_2 \cdot \boldsymbol{r}}{\|\boldsymbol{e}_2\|\,\|\boldsymbol{r}\|}. \tag{8}$$

The m pairs of the hard constraints (1) and (2) mean that each (x'_{i_j}, y'_{i_j}) must be the coordinates obtained by projecting $\boldsymbol{p}_{i_j}$ onto the new projection plane. The soft constraints (3), (4), and (5) imply that $\boldsymbol{e}'_1$ and $\boldsymbol{e}'_2$ should be orthonormal. The soft constraints (6), (7), and (8) indicate that $\boldsymbol{r}$ should be a unit vector, and that $\boldsymbol{e}'_1$ and $\boldsymbol{e}'_2$ should be the rotations of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ around $\boldsymbol{r}$. These vectors are depicted in Figure 1 in the three-dimensional manner, where we assume only one node is constrained (that is, $m = 1$).

We process these constraints by solving the following constrained least squares problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{6} w_i f_i^2 \\ \text{subject to} \quad & \|\boldsymbol{e}'_1\|^2 x'_{i_j} + (\boldsymbol{e}'_1 \cdot \boldsymbol{e}'_2)\, y'_{i_j} = \boldsymbol{p}_{i_j} \cdot \boldsymbol{e}'_1 \\ & (\boldsymbol{e}'_1 \cdot \boldsymbol{e}'_2)\, x'_{i_j} + \|\boldsymbol{e}'_2\|^2 y'_{i_j} = \boldsymbol{p}_{i_j} \cdot \boldsymbol{e}'_2 \end{aligned}$$

for $j = 1, 2, \ldots, m$, where

$$\begin{aligned} f_1 &= \|\boldsymbol{e}'_1\| - 1 \\ f_2 &= \|\boldsymbol{e}'_2\| - 1 \\ f_3 &= \frac{\boldsymbol{e}'_1 \cdot \boldsymbol{e}'_2}{\|\boldsymbol{e}'_1\|\,\|\boldsymbol{e}'_2\|} \\ f_4 &= \|\boldsymbol{r}\| - 1 \\ f_5 &= \frac{\boldsymbol{e}'_1 \cdot \boldsymbol{r}}{\|\boldsymbol{e}'_1\|\,\|\boldsymbol{r}\|} - \frac{\boldsymbol{e}_1 \cdot \boldsymbol{r}}{\|\boldsymbol{e}_1\|\,\|\boldsymbol{r}\|} \\ f_6 &= \frac{\boldsymbol{e}'_2 \cdot \boldsymbol{r}}{\|\boldsymbol{e}'_2\|\,\|\boldsymbol{r}\|} - \frac{\boldsymbol{e}_2 \cdot \boldsymbol{r}}{\|\boldsymbol{e}_2\|\,\|\boldsymbol{r}\|} \end{aligned}$$

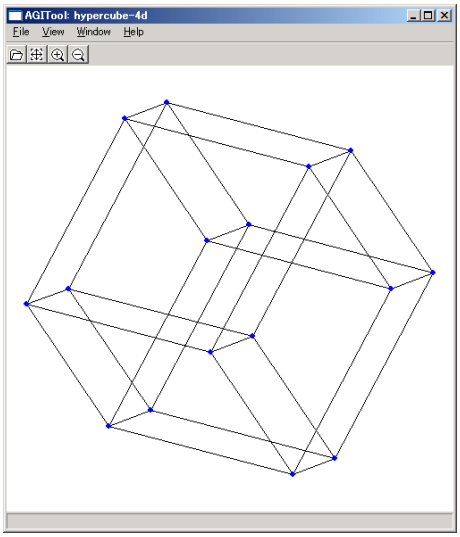


Figure 2: The prototype system AGI.

and each w_i is the nonnegative weight associated with f_i. In this problem, we use the hard constraints (1) and (2) as the actual constraints, whereas we embed the soft constraints (3), (4), (5), (6), (7), and (8) in the objective function by composing the weighted sum of the squares of their violations. This constrained least squares problem can be regarded as a two-level case of constraint hierarchies (Borning, Freeman-Benson & Wilson 1992), which are known as one of the most widely used frameworks for soft constraints (Barták 2002). For the weights, we simply use $w_i = 1$ for each i in our current implementation.

5 Implementation

Using the extended high-dimensional method presented in the previous section, we implemented a prototype graph drawing system called AGI[1] in C++. It adopts LAPACK 3.0[2] for linear computation including eigenvector calculation. To obtain graph-theoretic distances, it employs Floyd's algorithm (Ishihata 1989). To solve constrained least squares problems, it uses the C version of DONLP2[3]. For graphical user interfaces, it exploits the wxWidgets[4] multi-platform toolkit.

Figure 2 illustrates a screen shot of the AGI system. Given a file containing a graph in the GraphML[5] format, it displays a two-dimensional layout of the graph, and allows a user to interactively update the layout by dragging one of the nodes successively.

6 Experiments

To evaluate the extended high-dimensional method proposed in this paper, we performed experiments. For this purpose, we compiled the prototype system by using GCC 3.3.1 with the –O3 option, and executed it on a 1 GHz Pentium M processor running Windows XP.

Below we provide three examples of executing the extended method. The first one is the layout of the AT&T graph[6] `ug_45`, which consists of 97 nodes and 182 edges. Figure 3(a) shows the initial layout of the graph, whose internal dimensionality is 62. Suppose that we examine the detail of the left part of the graph. Figure 3(b) gives a graph layout after dragging a node in the left part, and Figure 3(c) illustrates a graph after dragging another node in the part. Note that the uppermost node in the left part stayed at the same position in Figures 3(b) and (c); its position was constrained to be constant by the extended high-dimensional method. The time required to compute the initial layout was 16 milliseconds, and the times needed to update the layouts were typically within 50 milliseconds.

The second example is the layout of the AT&T graph `ug_223`, which consists of 244 nodes and 340 edges. Figure 4(a) depicts the initial layout of the graph, whose internal dimensionality is 207. Suppose that we investigate the detail of the right part of the graph. Figures 4(b), (c), and (d) show the layouts obtained by successively dragging three nodes in different directions. The time needed to calculate the initial layout was 172 milliseconds, and the times taken to update the layouts were usually within 80 milliseconds.

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) illustrates its initial layout, whose internal dimensionality is 697. Figure 5(b) shows 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) give layouts after dragging other two nodes one by one. The time required to obtain the initial layout was 25.3 seconds, and the times needed to update the layouts were typically within 100 milliseconds.

7 Discussion

The force-directed approach can be easily extended to incorporate constraints (Di Battista et al. 1999, Tamassia 1998). It is straightforward to constrain nodes to be at fixed positions, and also it is possible to express more complex constraints in terms of attractive and repulsive forces of springs between nodes. However, it is difficult for the force-directed approach to efficiently obtain stable layouts of large graphs.

Our high-dimensional approach uses linear transformations to obtain two-dimensional graph layouts from high-dimensional ones. By contrast, fisheyeing often adopts nonlinear transformations to expand and emphasize details of visualized complex information (Sarkar & Brown 1994). It may be useful to introduce such a powerful nonlinear transformation into the high-dimensional approach, for which, however, we will probably need to trade off the efficiency in updating two-dimensional graph layouts.

8 Conclusions and Future Work

In this paper, we proposed an extended high-dimensional method for the interactive drawing of general undirected graphs. The method enables multiple graph nodes to be controlled simultaneously by introducing the notion of soft constraints as well as by using additional constraints on multiple controlled nodes.

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

[1] AGI stands for "Active Graph Interface."
[2] `http://www.netlib.org/lapack/`
[3] `http://plato.la.asu.edu/donlp2.html`
[4] `http://www.wxwidgets.org/`
[5] `http://graphml.graphdrawing.org/`
[6] `ftp://ftp.research.att.com/dist/drawdag/ug.gz`

Acknowledgments

We would like to thank the anonymous referees for their useful comments and suggestions. This research has been supported by the Kayamori Foundation of Informational Science Advancement.

References

Barták, R. (2002), 'Modelling soft constraints: A survey', *Neural Netw. World* **12**(5), 421–431.

Borning, A., Freeman-Benson, B. & Wilson, M. (1992), 'Constraint hierarchies', *Lisp Symbolic Comput.* **5**(3), 223–270.

Bruß, I. & Frick, A. (1996), Fast interactive 3-D graph visualization, *in* 'Graph Drawing—GD'95', Vol. 1027 of *LNCS*, Springer, pp. 99–110.

Di Battista, G., Eades, P., Tamassia, R. & Tollis, I. G. (1999), *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall.

Eades, P. (1984), 'A heuristic for graph drawing', *Congressus Numerantium* **42**, 149–160.

Gajer, P., Goodrich, M. T. & Kobourov, S. G. (2000), A multi-dimensional approach to force-directed layouts of large graphs, *in* 'Graph Drawing—GD2000', Vol. 1984 of *LNCS*, Springer, pp. 211–221.

Harel, D. & Koren, Y. (2002), Graph drawing by high-dimensional embedding, *in* 'Graph Drawing—GD2002', Vol. 2528 of *LNCS*, Springer, pp. 207–219.

Hosobe, H. (2004), A high-dimensional approach to interactive graph visualization, *in* 'Proc. ACM SAC', pp. 1253–1257.

Ishihata, K. (1989), *Algorithms and Data Structures*, Iwanami-Shoten. In Japanese.

Kamada, T. & Kawai, S. (1989), 'An algorithm for drawing general undirected graphs', *Inf. Process. Lett.* **31**(1), 7–15.

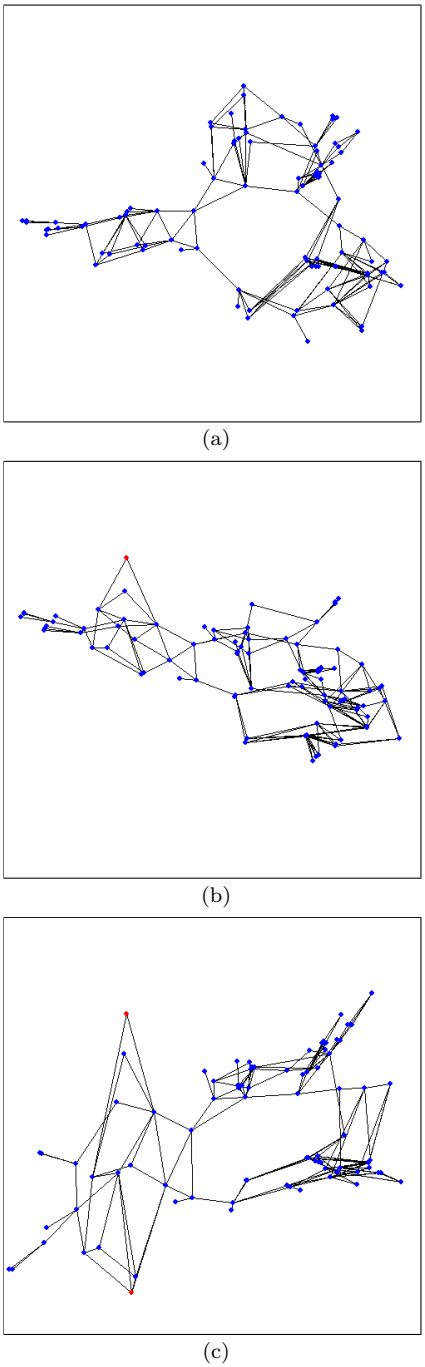
Kruskal, J. B. & Seery, J. B. (1980), Designing network diagrams, *in* 'Proc. 1st General Conf. on Social Graphics', U.S. Dept. Census, pp. 22–50.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (1992), *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edn, Cambridge Univ. Press.

Sarkar, M. & Brown, M. H. (1994), 'Graphical fisheye views', *Comm. ACM* **37**(12), 73–83.

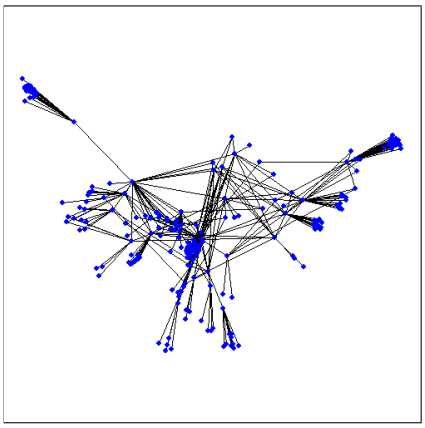
Tamassia, R. (1998), 'Constraints in graph drawing algorithms', *Constraints* **3**(1), 87–120.

Young, F. W. (1985), Multidimensional scaling, *in* 'Encyclopedia of Statistical Sciences', Vol. 5, Wiley, pp. 649–658.

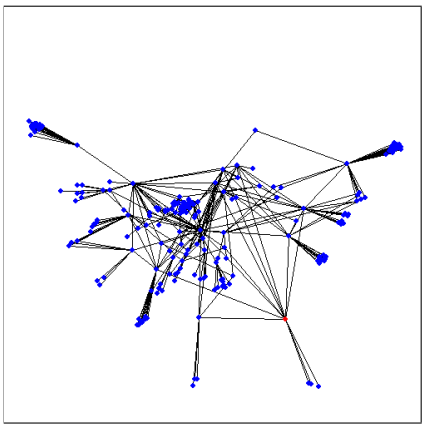


(a) (b) (c)

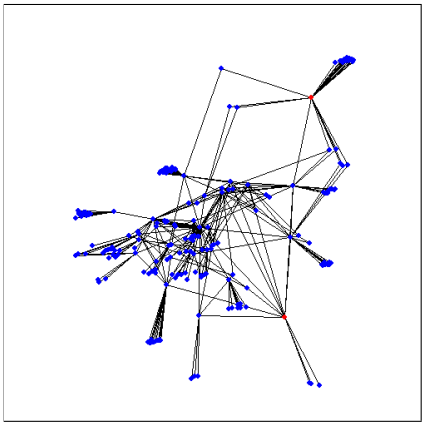
Figure 3: Interactive layout of the AT&T graph ug_45 (with 97 nodes and 182 edges).



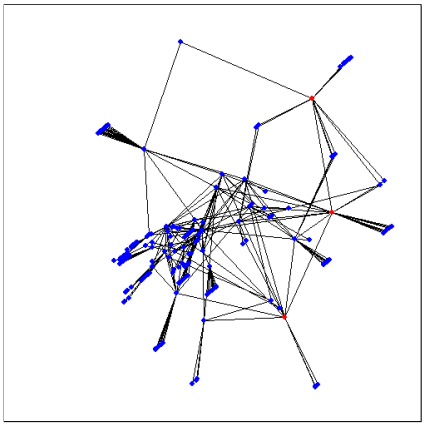
(a)



(b)

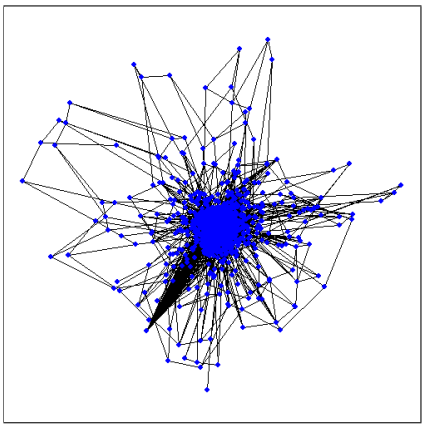


(c)

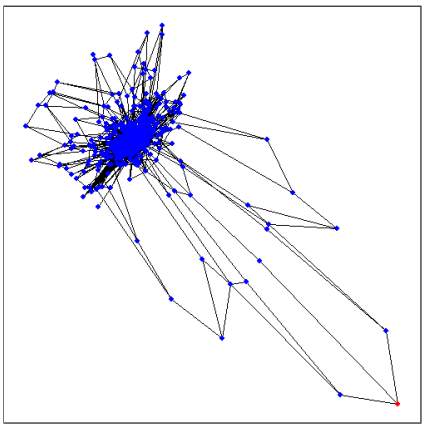


(d)

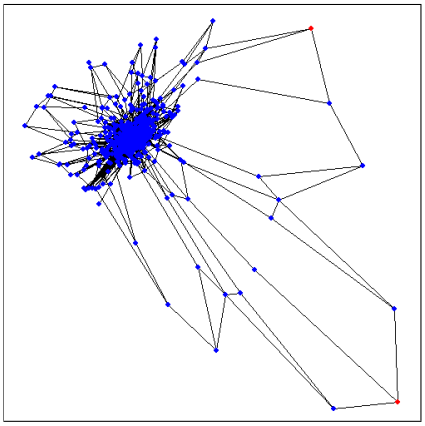
Figure 4: Interactive layout of the AT&T graph `ug_223` (with 244 nodes and 340 edges).



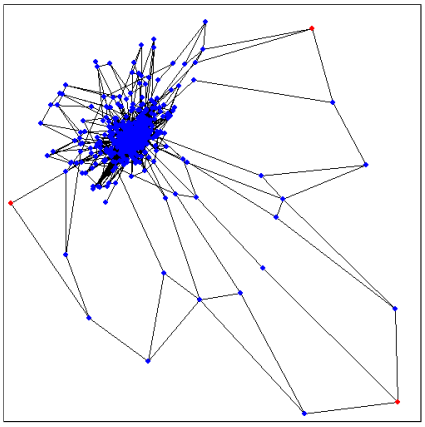
(a)



(b)



(c)



(d)

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