

Exploring the Webpages in Visualization

Han Xiao

*School of Information Engineering, Beijing University of Posts and Telecommunications, China
artex.xh@gmail.com*

Abstract

We propose a visualization method based on Isomap and force displacement for linked discrete data such as a set of linked webpages. Unlike conventional visualization methods which study the link structures and textual contents separately, We take advantage of the relationship between the links and the content semantics to refine the graphical layout. In our approach, high-dimensional documents are first mapped to two-dimensional Euclidean space using Isomap, we then design a Force-directed algorithm to rearrange the coordinates of the vertices, i.e. use links as leverage to adjust the distance between the vertices in the semantic Euclidean space. The visualization experiment we conducted shows that the proposed method is elegant, conceptually intuitive and highly efficient.

1. Introduction

Today, information visualization in scientific research is generally applied to the large-scale collections of non-numerical information. Information visualization focused on the creation of approaches for conveying abstract information in intuitive ways. It takes advantage of the human eye's broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once. It also builds a direct interface between user and machine. The importance of visualizing documents is increasing since documents such as web pages, blogs, E-mails, patents and scientific articles are being accumulated rapidly. When working with large volumes of data like these, visualization is a useful tool for us to understand the complex and high dimensional data, and it enables us to browse intuitively through huge amounts of data. However, the seeks to more significant layout is still an open problem. How to represent a large amount of data in a small space? How to organize the visual space sufficiently and provides some meanings? How to manipulate perceptual interface to capture the latent

relationships between data? In this paper, we will answer these three questions with a representative case on webpages. Webpages can be seen as high-dimensional data with link structure. Each page can be described in vector space model, its values usually formulated as a function of tokens frequency. The link dependencies convey the specific type of relationship between pages. A group of linked pages are likely similar on semantics (e.g. "Related News"). To visualize the webpages, we proposed an approach called Force-directed Isomap to induce the semantics relations and link structures simultaneously in two-dimensional Euclidean space.

The remainder of the paper is organized as follows: In Section 2, we review related work. In Section 3, we formulate the Force-directed Isomap in detail. We then demonstrate the effectiveness of our method by visualizing document on HPWEB data set in Section 4. Finally, we present concluding remarks and a discussion of future work in Section 5.

2. Related works

The basic problem of visualizing data is: finding meaningful low-dimensional structures hidden in their high dimensional observations. Although the input dimensionality may be quite high, the perceptually meaningful structure of these data has many fewer independent degrees of freedom. This process is called dimensionality reduction. A number of methods for this problem have been proposed, such as principal component analysis (PCA) [1], multidimensional scaling (MDS) [2] and isometric feature mapping (Isomap) [3]. PCA and MDS, are simple to implement, efficiently computable, and guaranteed to discover the true structure of data lying on or near a linear subspace of the high-dimensional input space. PCA finds a low-dimensional embedding of the data points that best preserves their variance as measured in the high-dimensional input space. Classical MDS finds an embedding that preserves the interpoint distances, equivalent to PCA when those distances are Euclidean. Isomap solving dimensionality reduction problems that

uses easily measured local metric information to learn the underlying global geometry of a data set. Unlike classical techniques such as principal component analysis (PCA) and multidimensional scaling (MDS), Isomap is capable of discovering the nonlinear degrees of freedom that underlie complex natural observations.

In text processing, some semantic models can also be used to extract a low-dimensional representation of a document by analyzing relationships between a set of documents and the terms they contain. Latent semantic analysis (LSA) [4] is a method for finding a low-rank approximation to the term-document matrix, and enables document comparison in the concept space. Probability latent semantic index (PLSI) [5] and Latent Dirichlet allocation (LDA) [6] are well-known topic models for analyzing documents and other discrete data. In these hierarchical probabilistic models, a document is modeled as a multinomial on topics, where each topic is modeled as a multinomial on words. However, they are not appropriate for visualization since the representation is an embedding in the simplex space but not in the Euclidean space. As a probability distribution over words, each document in the text collection can be represented as a point on the simplex. Similarly, each topic can also be represented as a point on the simplex. As shown in Figure 1, the topics span a low dimensional subsimplex and the projection of each document onto the low-dimensional subsimplex can be thought of as dimensionality reduction.

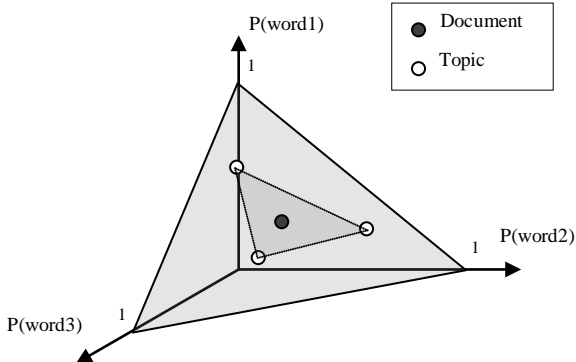


Figure 1. With a vocabulary containing W distinct word types, a W dimensional space can be constructed where each axis represents the probability of observing a particular word type. The $W-1$ dimensional simplex represents all probability distributions over words.

Though much work has been done, the issues we focus on in this work (i.e. integrate semantics and link structures to refine the graphic layout) have not been sufficiently investigated. Our method addresses this problem holistically and efficiently.

3. Force-directed Isomap

In the problem of webpages visualization, the textual content on each page can be first described as a high-dimensional vector. We then apply the Isomap algorithm to find low dimensional structure in high-dimensional data points. Similar pages which have latent semantic relations will be clustered into the same region, while those unrelated pages will be placed far apart. Isomap provides a simple method for estimating the intrinsic geometry of data manifold based on a rough estimate of the neighbors of each data point on the manifold. The simplicity of Isomap, in contrast to the singular value decomposition used in LSA, makes it highly efficient and generally applicable to a broad range of data sources and dimensionalities, and makes possible on large volume data analysis and visualization.

The Isomap algorithm takes as input the distances $d_x(i, j)$ between all pairs i, j from N data points in the high-dimensional input space X , measured in the standard Euclidean metric. The algorithm outputs coordinate vectors (x, y) in a two-dimensional Euclidean space that best represent the intrinsic geometry of the data. The complete Isomap algorithm used in our paper has four steps, which are detailed in Table 1.

Table 1. Isomap Algorithm used in this paper.

1. Compute distance in original space
Given a set of N pages, each of them is represented as a M -dimensional vector. Compute pairwise distance measured in the standard Euclidean metric.
2. Construct neighborhood graph
Define the graph G over all data points by connecting points i and j if [as measured by $d_x(i, j)$] i is one of the K nearest neighbors of j (K -Isomap). Set edge lengths equal to $d_x(i, j)$.
3. Compute shortest paths
Initialize $d_g(i, j) = d_x(i, j)$ if i, j are linked by an edge; $d_g(i, j) = \infty$ otherwise. Applied Floyd's algorithm on graph G : for each value of $k = 1, 2, \dots, N$, replace all entries $d_g(i, j)$ by $\min\{d_g(i, j), d_g(i, k) + d_g(k, j)\}$. The matrix of final values $D_g(i, j)$ will contain the shortest path distances between all pairs of points in G .
4. Construct two-dimensional embedding
Let λ_1, λ_2 be the two largest eigenvalue (in descending order) of the matrix $\tau(D_g)^{-1}$, and v_p^i be the i -th component of the p -th eigenvector. Then set the two-dimensional coordinate of data i equal to $(\sqrt{\lambda_1} v_1^i, \sqrt{\lambda_2} v_2^i)$.

1. The operator τ is defined by $\tau(D) = -HS / 2$, where S is matrix of squared distances $\{S_{ij} = D_{ij}^2\}$, and H is the centering matrix $\{H_{ij} = \delta_{ij} - 1 / N\}$.

The only free parameter K appears in Step 2. Selecting a suitable value for K is not difficult since Isomap is topologically stable over a range of neighborhood sizes [3]. Note that in step 3 the basic Floyd's algorithm requires $O(n^3)$ operations, a paralleled version is given in [7]. Throughout the procedure described in Table 1, we found that each step can be parallelized, which makes the whole algorithm more efficiently in a parallel computing environment.

After the above step, the webpages have been placed into two-dimensional Euclidean space by Isomap. But some problems in visual representation still remain. Firstly, some high-dimensional data are mapped to the same coordinate value although they are actually different in the original space. This is inevitable in dimensional reduction, which we call "collapsed graph", often result inability to visual discerns. Secondly, the links present in pages have not been visualized. An intuitive method may be as simple as adding extra edges between the linked data points. However, this kind of drawing is somewhat arbitrary since it does not take the valuable insights between links and semantics into account.

We propose an approach based on Force-directed algorithm to solve these two problems simultaneously. Force-directed algorithms [8] are a class of algorithms for drawing graphs in an aesthetically pleasing way. It achieves this by assigning forces amongst the set of edges and the set of nodes; the most straightforward method is to assign forces as if the edges were springs and the nodes were electrically charged particles. The entire graph is then simulated as if it were a physical system. The forces are applied to the nodes, pulling them closer together (Hooke's law¹) or pushing them further apart (Coulomb's law²). This is repeated iteratively until the relative positions of the nodes do not change anymore. The physical interpretation of this equilibrium state is that all the forces are in mechanical

equilibrium and hence a good layout for the graph is found. Taking advantage of the flexibility of Force-directed algorithm, we can introduce forces other than mechanical springs and electrical repulsion; examples include logarithmic function, exponential function or simply linear function.


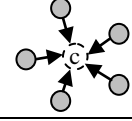
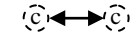
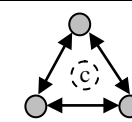
The classical Force-directed algorithms, however, do not constrain positions of vertices and the relative distance between them. The graph will be freely stretched out in space eventually, in which all the edges are of more or less equal length and there are as few crossing edges as possible.

Inspired by the assumptions that linked pages are likely to be similar in semantics, we remodel the force based on the following guidelines:

- (1) Vertices repel each other if they shared almost the same coordinate to prevent the graph from collapsing.
- (2) Vertices that are linked together attract each other, since they may have semantic relationship.
- (3) The relative positions of vertices should be maintained or fixed as much as possible, otherwise the meaningful low-dimensional structures and semantic distance found by Isomap will be lost.
- (4) The forces, either attractive or repulsive, only act on the neighborhood vertices; if the distance between two vertices is above a certain threshold, there are no interacting forces between them.

These rules incorporate the link structure information into the visualization space and refine the semantic similarities between pages with the help of links. When doing so, they maintain the semantic consistency in the Euclidean space, thus make the visualization more appealing. The forces used in our algorithm are given in Table 2. The "core" is defined as the collapsed point, on which more than one vertices shared almost the same coordinate value.

Table 2. Four kinds of forces defined and used in our algorithm

Forces	Descriptions	Illustrations	Functions ($0 < x < 1$)
F_1 (attractive)	The two-way attractive force between linked vertices		$k_1 \text{Beta}(2, 3) \cdot x^2 \cdot (1-x)^3$
F_2 (attractive)	The one-way attractive force between the vertices and their corresponding "core"		$k_2 x$
F_3 (repulsive)	The two-way repulsive force between every two "cores"		$k_3 / -\log(1-x)$
F_4 (repulsive)	The two-way repulsive force between every two vertices which belong to the same "core"		$k_4 \text{Beta}(2, 3) \cdot x^2 \cdot (1-x)^3$

¹. $F = -kx$, where k is the force constant (or spring constant).

². $F = k_e q_1 q_2 / r^2$, where k_e called Coulomb's constant.

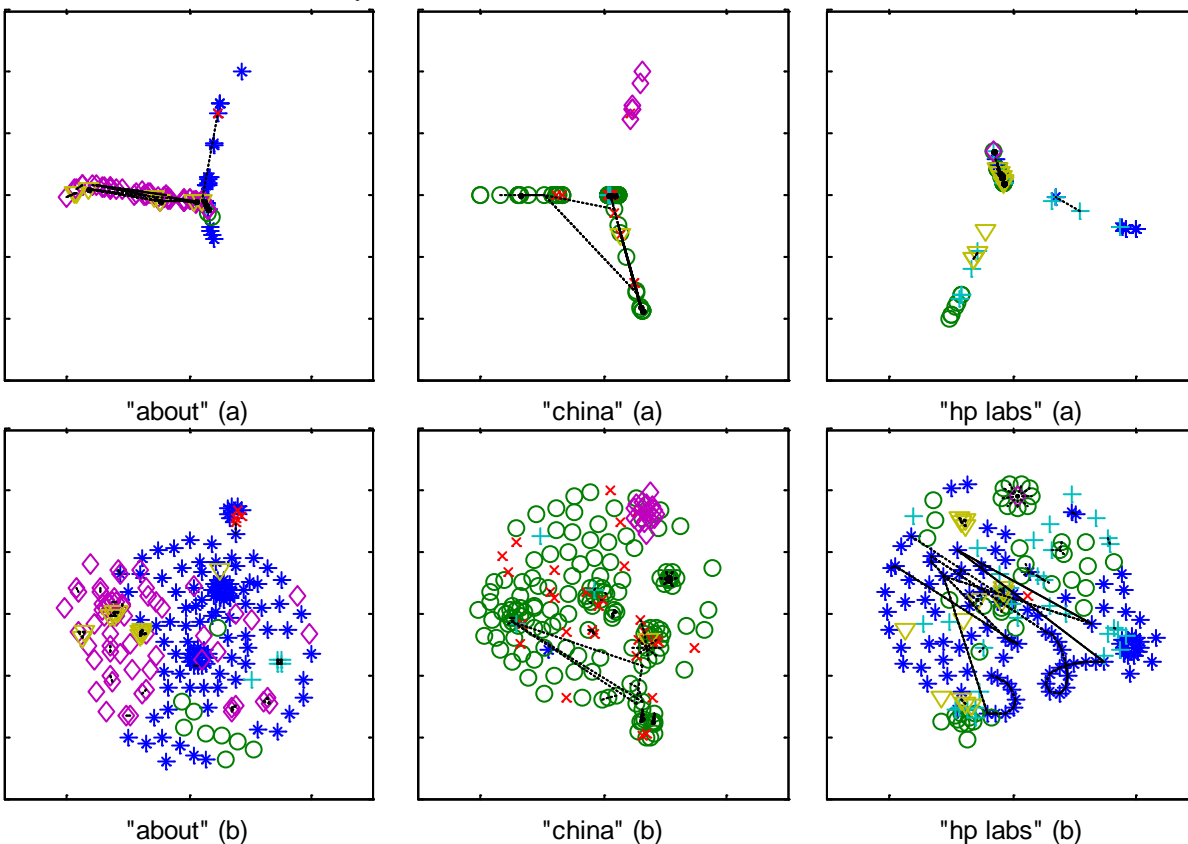
The configuration for the number of cores and their initial values are crucial factors affecting performance. Although the cores can be simply set to the distinct coordinates that output from Isomap, this result in too many cores with short distance, they will repulse each other and derange the semantic relations arbitrarily. In this paper, we apply k-means algorithm [9] to find cores. Given the data set $\{(x_1, y_1), \dots, (x_N, y_N)\}$ consisting of the coordinates given by Isomap, our goal is to partition the data set into a number of clusters and set the centers of the clusters to the initial coordinates of the cores. One notable feature of the K-means algorithm is that some of the clusters are empty at the end of iterations. It's impossible to choose the best number of cores at the beginning, so we use n_{cMAX} to constrain the maximum number of cores instead.

Pseudo-code for the algorithm is given in Appendix A. The initial configurations of the vertice positions are specified by the output of Isomap. There are two steps in each iteration: calculate the effect of four kinds of force on each vertex and "core" given in Table 2, and then limit the total displacement by the temperature. A special case occurs when vertices are in the same position: our implementation acts as though the two vertices are a small distance apart in a randomly chosen orientation: this leads to a violent repulsive effect separating them. The displacement of a vertex is limited to some maximum value in each iteration by annealing algorithm. And this maximum value decreases over time; so, as the layout becomes better,

the amount of adjustment becomes finer and finer. Different cooling schedules and different numbers of iterations may result in different graphs. In our implementation, the temperature simply decays to 0 in a linear fashion.

4. Experiments

We used HPWEB data set for our experiment on visualization. It consist of about 50 thousand webpages from Hewlett-Packard external web. We parsed all webpages and tokenized the words, and then omitted stop-words and words that occurred fewer than 100 times. The vocabulary size was 34036. Thus, each page is represented in a 34036-dimensional vector. The weighing is simply the term frequency. We did a full-text index by Lucene on dataset. Given a query, we use the top 200 pages retrieved from Lucene and analyze the link relations between them. Our force-directed Isomap is to draw these query results in a graph. As an evaluation measurement, these 200 pages are clustered into 6 classes with the K-means algorithm. Differs from K-means used to find cores, this clustering is in the original document space. The accuracy and elegancy of the graph could be intuitively measured: pages in the same class, or linked to each other, should be placed close together, but not collapsed to the same point; while pages in different class should be placed far away from each other in the visualization space.



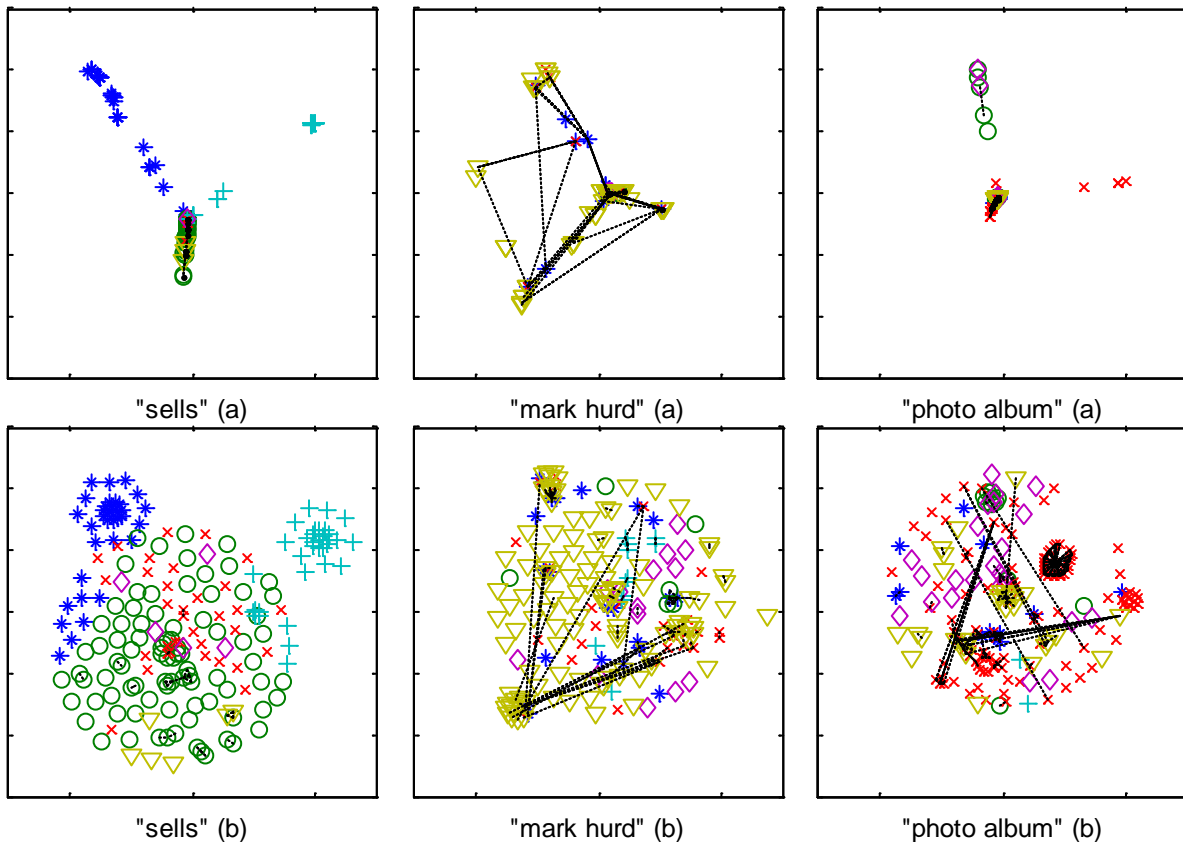


Figure 2. Visualization result of Isomap [label with (a)] and Force-directed Isomap [label with (b)] on 6 queries. The neighbor size K in Isomap set to 5, the parameters used in Force-directed algorithm are: $n_{cMAX}=10$; $k_1=1$; $k_2=1600$; $k_3=100$; $k_4=0.5$; $T=1$; $\lambda=1.1$.

Figure 2 shows visualization results obtained by original Isomap algorithm in contrast with Force-directed Isomap. Here each point represents a page, and different markers with color represent different labels. Links between pages are drawn in black dashed line. The original Isomap gives a collapsed layout, thus most of the points are located at same position. In our experiments on 10 different queries, the Isomap method gives 50 distinct coordinates for 200 pages on average, more than 70% of the pages' positions are coincidence on two-dimensional Euclidean space. Due to this collapsed layout, link relations cannot be sufficiently presented. The Force-directed Isomap gives an aesthetically-pleasing layout as pages may repel or attract each other to avoid collapsing, meanwhile pages in the same class are likely to be clustered together. By applying different force functions in our algorithm, the graph layout is well-distributed globally and regularly shaped locally. The computational time of Force-directed Isomap on a PC with 2GHz Xeon CPU and 4GB memory is on average

5 seconds for a query with 200 pages. A parallel version of algorithm will be faster.

Another notable feature of Force-directed Isomap is that it captures the link structures and represent them topologically even in semantic space. Figure 3 shows some case-studies obtained by Force-directed Isomap. Typical examples are show in the right part of Figure 3, the link structures embedded in pages extend their topological shape in space constrainedly and elegantly, without disrupt the semantic consistency. "Mailing list" pages contains a home page that link to others, thus output a star-shaped layout. "DSCxxxx" which actually is photo album, only have "last photo" and "next photo" hyperlinks on each page, make them only connect to their adjacent neighbors, lead a linear layout in visualization. The pages which title are "Armchair CEO" come from a blog, the template is designed to help visitors navigate or switch panels conveniently, so that each page is linked with any other. Visualization result of this kind of well-linked pages is complete graph (Figure 3.h) or quasi-complete graph (Figure 3.f)

5. Conclusions

In this paper, we proposed a visualization method based on Isomap and Force displacement, Force-directed Isomap, for linked data such as webpages. We have confirmed experimentally that Force-directed Isomap can visualize textual and link relations between pages simultaneously in two-dimensional Euclidean space. The result layout of our algorithm is well-distributed on vertices and regularly on link structure. The simplicity of implementation and fast convergence also show that our data visualization approach will become a useful and practical tool for visualizing documents and expanding cognitive resources.

Future work will be focused on extending Force-directed Isomap with more link information between pages, and use statistical approach to estimate the parameters in force function. By designing more flexible force function that integrated textual information, edge-lengths, edge-crossings and link transitivity, the visualization result is expected to be more helpful in knowledge representation and exploration.

6. Acknowledgement

We thank Xu Pu for useful conversations, Yuhong Xiong and Xiaojie Wang for helpful comments on a previous draft, and Hewlett-Packard Laboratories for HPWEB dataset.

References

- [1] H. Murase, S.K. Nayar, Visual learning and recognition of 3-D objects from appearance, *International Journal of Computer Vision*, 14, 5, 1995.
- [2] W. Torgerson. *Theory and methods of scaling*. Wiley, New York, 1958.
- [3] J.B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500): pp. 2319-2323, 2000.
- [4] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer and R Harshman. Indexing by latent semantic analysis, *Journal of the American society for information science*, 1990
- [5] T. Hofmann. Probabilistic latent semantic analysis. In UAI'99: Proceedings of 15th Conference on Uncertainty in Artificial Intelligence, pp. 289-296, 1999.
- [6] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3: pp. 993-1022, 2003.
- [7] V. Kumar, A. Grama, A. Gupta and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, pp. 257-297.
- [8] T.M.J. Fruchterman, E.M. Reingold, Graph Drawing by Force-directed Placement, *Software: Practice and*

Experience, VOL.21(1 1), pp. 1129-1164, 1991

- [9] C. Ding, X.J. Wang. "K-means Clustering via Principal Component Analysis". In Proceedings of International Conference of Machine Learning (ICML 2004), pp 225-232. July 2004.

Appendix

A. Pseudo-code of Force-directed Isomap

```
// V is vertices, E is edges/links and C is cores
G:=(V,E,C);
set up initial vertex's positions to the output of Isomap
set up initial core's positions to the distinct values of V
set up core's members to {v} that at same position
loop
  energy:=0;
  //calculate the two-way attractive force F1
  for each e in E
    //each edge is an ordered pair of vertices e=(v,u)
    Δ:=e.v.pos-e.u.pos;
    if (0<|Δ|<1)
      e.v.disp:=e.v.disp-(Δ/|Δ|)*F1(|Δ|);
      e.u.disp:=e.u.disp+(Δ/|Δ|)*F1(|Δ|);
    end
  end
  //calculate the one-way attractive force F2
  for each c in C
    //each core has a set of vertices {v}
    for each v in c.{v}
      Δ:=c.disp-c.v.disp;
      if (0<|Δ|<1)
        c.v.disp:=c.v.disp+(Δ/|Δ|)*F2(|Δ|);
      end
    //calculate the two-way repulsive force F3
    for each c in C
      for each d in C
        if (d≠c)
          Δ:=c.disp-d.disp;
          if (0<|Δ|<1)
            c.disp:=c.disp+(Δ/|Δ|)*F3(|Δ|);
          end
        //calculate the two-way repulsive force F4
        for each c in C
          for each v in c.{v}
            for each u in c.{v}
              if (v≠u)
                Δ:=c.v.disp-c.u.disp;
                if (0<|Δ|<1)
                  c.v.disp:=c.v.disp+(Δ/|Δ|)*F4(|Δ|);
                end
              //limit the maximum displacement to the temperature T
              //update the position of vertices and cores
              for each v in V
                v.pos:=v.pos+(v.disp/|v.disp|)*min(|v.disp|,T);
                energy:=energy+|v.disp|;
              end
              for each c in C
                c.pos:=c.pos+(c.disp/|c.disp|)*min(|c.disp|,T);
                energy:=energy+|c.disp|;
              end
            //cool down the temperature to adjust finer in next
            iteration
            T:=T/λ
          until T≈0 or energy<ε
```