Visualizing the embedding of objects in Euclidean space

Michael Littman, Deborah F. Swayne, Nathaniel Dean, and Andreas Buja
Bellcore
Morristown, NJ 07962-1910

ABSTRACT

Matrices representing dissimilarities within a set of objects are familiar in mathematics, statistics and psychology. In this paper we describe XGvis, a software system which accepts diverse input data, such as graphs and multivariate data, develops a dissimilarity matrix from the data, and then iteratively and interactively embeds objects in a Euclidean space of arbitrary dimension. Using a technique called multidimensional scaling, objects are positioned so that their pairwise distances match the target dissimilarities as well as possible. Users can interact with XGobi, a software system for visualizing high-dimensional data, to browse the resulting embeddings. Mathematicians and statisticians have found XGvis to be useful for discovering and exploring structure.

XGvis runs under the X Window System™.

1. Introduction and motivation

A dissimilarity matrix D is a matrix in which $D_{i,j}$ represents the degree of dissimilarity between each pair of objects i and j. The row objects and the column objects are usually the same and dissimilarity is assumed to be a symmetric relation. Matrix D is then a square symmetric matrix, $D_{i,j} = D_{j,i}$, with zeros on the diagonal and non-negative entries everywhere else.

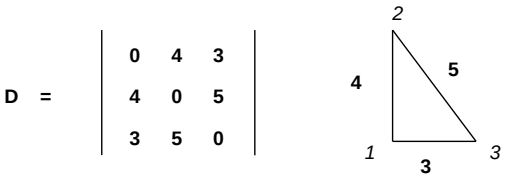

Figure 1: A dissimilarity matrix and its graphical interpretation.

The matrix in Figure 1 tells us how dissimilar 3 objects are from one another. Object 1, for instance, differs from object 2 by a score of 4. Construing similarity as closeness and dissimilarity as distance gives D a natural graphical interpretation. The Euclidean distances among three objects arranged in the right triangle shown in Figure 1 precisely match the dissimilarity matrix D.

In general, a geometric interpretation for a dissimilarity matrix D is created by representing each object by a point in some k dimensional space such that the distance between two objects i and j approximates $D_{i,j}$ as well as possible.

The process of creating these graphical representations is known as multidimensional scaling (MDS) and has been studied for many years. Recently, interactive tools for visualizing these representations have become accessible. In this paper, we describe a system called XGvis which combines procedures for computing and displaying distance embeddings with a collection of functions for generating input dissimilarity matrices from several data sources.

How might such a system be useful? Dissimilarity information is very common in mathematics, statistics and psychology. Potential sources include shortest path distances among vertices in a graph, measured dissimilarity among test objects in a psychometric experiment, and distances among cases in multivariate data.

The use of XGvis for graph theory applications is driven by another visualization tool developed at Bellcore. NETPAD (Dean et al., 1991) is an interactive program for creating graphs and graph layouts and analyzing these structures with a library of algorithms. NETPAD is capable of producing many interesting two dimensional layouts, and when NETPAD and XGvis are used together, they can create and display layouts of graphs in three or more dimensions. To the degree that a higher dimensional layout better captures interesting aspects of a graph's structure, it can lead to special insights into properties of the graph. These insights can be helpful in a mathematician's search for conjectures and counter-examples.

Dissimilarity information can also serve as an intermediate step between a high dimensional collection of data and a lower dimensional view of it. By computing the distance between pairs of points in a ten dimensional space and fitting these distances with objects in a three dimensional space, one can often get a more intuitive

X Window System is a trademark of MIT.

feel for the original data. This process is called dimension reduction and is widely used in data analysis and visualization. XGvis is well suited for this task.

There are some cases when a dissimilarity matrix itself is the raw data to be viewed. It is a fairly common technique in experimental psychology to extract similarity judgments from human subjects about a collection of objects. These similarity judgments are easily transformed to dissimilarities and viewed using XGvis.

By molding diverse input data into the form of a dissimilarity matrix and creating graphical views of the dissimilarity information, XGvis has proven to be a powerful tool for people in many disciplines.

2. The Method

XGvis accepts various types of input data and develops a dissimilarity matrix from each. It then uses a form of multidimensional scaling to iteratively determine an embedding for the data.

2.1. Handling of different data types

XGvis handles three primary data types: graphs, multivariate data and dissimilarity judgments. A dissimilarity matrix is developed from each of these using either a default method or one chosen by the user.

A graph consists of a set of objects called vertices and edges where vertices can be represented as points and edges as lines connecting pairs of points. For example, vertices could represent cities and edges could connect pairs of cities that have a direct flight between them. Graphs are used to represent such diverse systems as telephone networks, molecules and patterns of bibliographic references.

XGvis accepts graphs in several forms. The simplest is as a list of edges. Edges in XGvis are undirected – that is, if i is connected to j then j is assumed to be connected to i. A user can also optionally indicate a set of initial locations for the vertices. If these are omitted, vertices are placed at random.

The default approach for creating a dissimilarity matrix from the graph is to compute, for all pairs of vertices in the graph, the number of edges on the shortest path between them. XGvis uses a dynamic programming approach for finding all the shortest paths quickly.

When multivariate data is passed to XGvis, a dissimilarity matrix is created by computing the Euclidean distance between each pair of cases in the raw data. The $D_{i,j}$ cell of the matrix is the distance between case i and case j. Non-Euclidean metrics are computed at the user's request.

XGvis can accept a dissimilarity matrix directly. These matrices are assumed to be symmetric. In the event that a non-symmetric (though square) matrix is passed in, XGvis' default action is to create a symmetric matrix from it by computing the elementwise mean between the matrix and its transpose. The resulting matrix is symmetric and can be used by multidimensional scaling for creating a graphical layout.

2.2. Multidimensional scaling

Multidimensional scaling has a long and interesting history. The defining goal of multidimensional scaling (MDS) is to take an $n \times n$ dissimilarity matrix D and to produce vectors $x_1, \ldots, x_n$ such that the computed distance between every x_i and x_j approximates $D_{i,j}$. Up to 1964, people used classical or Torgerson-Young MDS (Torgerson, 1958) in which the dissimilarities are converted to an inner-product matrix and an eigenanalysis is performed.

In the early 60s, an alternate view of MDS began to emerge. The seminal contributor was Shepard (1962) who showed that to construct a faithful embedding it is sufficient to know the ranks rather than the actual values of the dissimilarities. That is, if for every two pairs (a,b) and (c,d) of objects it is known whether a is closer to b than c is to d, then a faithful embedding can be constructed. Shepard gave evidence that this is possible and Kruskal (Kruskal, 1964a, 1964b) devised an optimization criterion which incorporates this idea. Kruskal defined a measure of fit called "stress" (S).

$$S(D, x, f) := \frac{\sum_{i,j} \left[f(D_{ij}) - \|x_i - x_j\| \right]^2}{\sum_{i,j} \|x_i - x_j\|^2} \qquad (2.1)$$

Given a current configuration of the objects $x = x_1, \ldots, x_n$, the target dissimilarity matrix D and a monotonically nondecreasing function f, S is a measure of how poor the current embedding is. It is a normalized residual sum of squared differences between each $f(D_{i,j})$ and the distance from x_i to x_j. Kruskal provided a procedure for minimizing the stress by computing partial derivatives and applying the classical steepest descent method.

Kruskal's MDS based on this criterion with optimization of f over *all* monotonically nondecreasing functions is called "nonmetric MDS." It is called "nonmetric" since the actual values D_{ij} are thrown away by transforming them, while their ranks and *only* their ranks are retained because of monotonicity of the function.

The version of MDS used in XGvis is a metric relative of Kruskal's MDS. It optimizes x but leaves the function

f for the user to choose interactively from the set of powers of D.

Solving for the x_i's in metric Kruskal MDS is a sticky non-linear optimization problem. It does not boil down to an eigendecomposition such as the one used in Torgerson-Young MDS. Metric Kruskal is a more powerful method than classical Torgerson-Young scaling.

Unlike performing an eigenvalue decomposition, there does not seem to be a closed form algorithm for finding a metric Kruskal MDS embedding. Instead we use an iterative algorithm for finding the x_i's which minimize S.

The algorithm starts with some initial layout, which may be random, and minimizes S through a series of gradient descent moves on the point locations. With each step, the program moves every point a small amount in the direction which causes the stress to decrease maximally.

Aside from problems with local minima, easily avoided by using a large step size, the points settle to a reasonable layout quickly.

Visualizing an iterative algorithm offers two main benefits. First, it is possible to observe changes in the optimization behavior that might result from changes in the parameters of the stress function. For instance, one could experiment with an additional penalty term for constructing constrained layouts. Second, a user can find a good layout visually without having to construct a stopping criterion such as "run for x steps" or "run until the error drops below ϵ." An iterative method allows a user to implicitly use the criterion, "run until it looks right."

We have found this particularly important for setting the step size parameter. Larger steps help MDS make fast progress early and skip over local minima. Later in the run, a smaller step size is needed to let MDS settle on a solution. Terms like *larger*, *later* and *smaller* are very much problem dependent but are easily determined (at least in 2 and 3 dimensional embeddings) by watching the points move.

Interactive MDS optimization was implemented by one of us on a workstation in 1982, but it was never documented other than in a film (Buja, 1982).

3. Control and Visualization

Embeddings in XGvis proceed as follows: a user begins an XGvis session by invoking the program with some initial data. Using default routines, XGvis creates a dissimilarity matrix from the data. The user is then presented with a graphical interface for controlling the progress of MDS and manipulating the view of the current embedding.

Typically, the user will set some MDS parameters and then "run" MDS. The points are visible in a display window while MDS moves them. When the user is satisfied with the layout, MDS can be stopped and the resulting embedding examined with an integrated version of the XGobi visualization program (Swayne et al., 1991b, 1991c).

All interactions take place in two windows on the workstation or personal computer screen. The XGvis control panel window is used to start, stop and restart MDS as well as to control the MDS-related parameters. The XGobi display window is used to watch the progress of the optimization and to interact in various ways with the view of the data or figure.

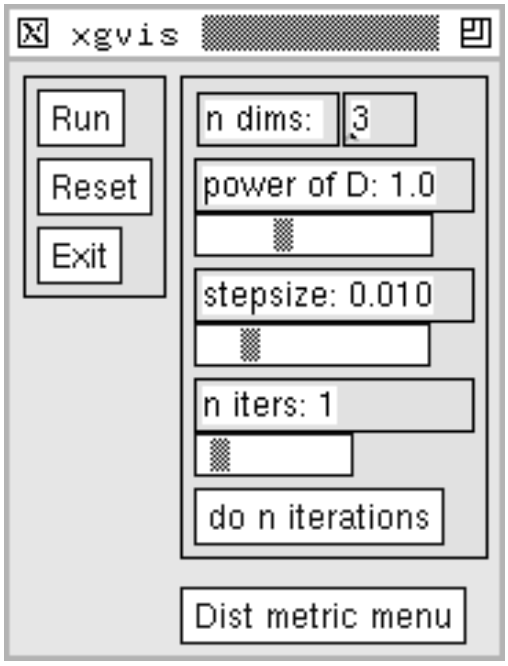

Figure 2: XGvis Control Panel

3.1. Control panel

The XGvis control panel is shown in Figure 2. The leftmost subwindow contains a *Run* button that allows a user to start and stop the optimization and a *Reset* button that returns the data or figure to its original configuration. The labels, buttons and scrollbars in the rightmost windows allow a user to adjust various optimization and scaling parameters. These parameters can be set before MDS begins or while it is running; they are briefly described here.

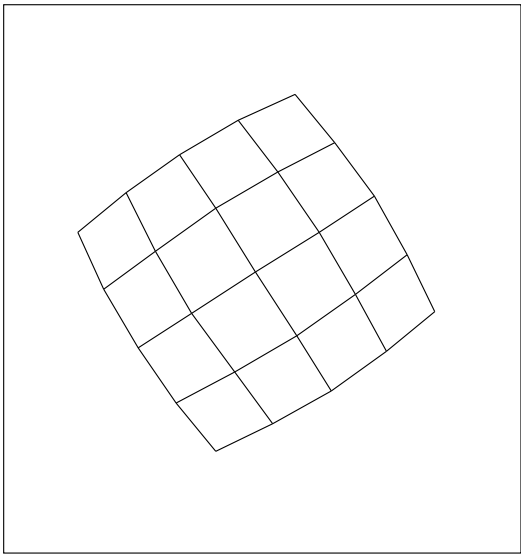

Figure 3: Grid, power of $D = 0.5$

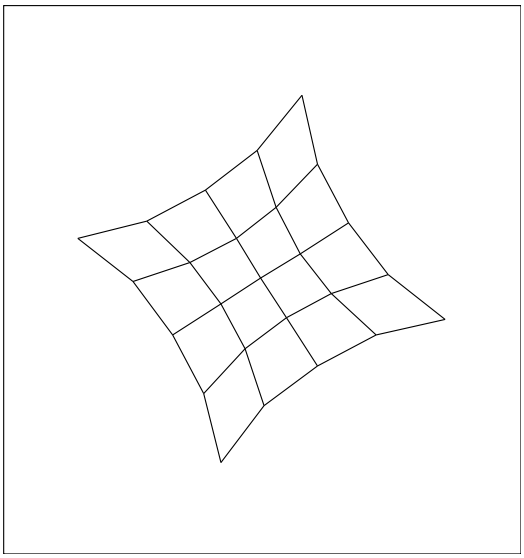

Figure 4: Grid, power of $D = 2.0$

Dimension: By typing an integer into the text window at the right of the label *n dims*, the user specifies the dimension of the Euclidean space in which to embed the point cloud or figure. Reasonable choices are 2 (XGvis embeds the points or objects in the plane), 3 (XGvis constructs a three dimensional layout), or up to 5 or 6, in which case special techniques and imagination are required to interpret the resulting structures. The interface actually permits embeddings in up to twenty dimensions but we have not found use for nearly that many. Occasionally one dimensional embeddings (all points on a line – unidimensional scaling) are interesting and they are also supported.

Transformation: The user adjusts the scrollbar beneath the label *power of D* to select the power to which each element of D should be raised. In this way, XGvis allows some variation in the function f shown in Equation 2.1. We restrict $f(D_{i,j})$ to $D_{i,j}^p$ where p is the value read from the scrollbar. Typically, a value of $p = 1.0$ is the most meaningful: this tells XGvis to fit the dissimilarities in D directly.

Striking and useful effects can be obtained by changing p from its default value. Small values of p tend to minimize differences in the dissimilarity matrix, often resulting in a "rounder" or more compact layout. A 25 node graph connected as a 5×5 grid appears perfectly square when displayed with $p = 1.0$, but note the rounded appearance of the 5×5 grid as it is shown in Figure 3: this embedding was produced using $p = 0.5$. In the limit $p = 0.0$ and MDS tries to make all pairs of objects distance 1 from each other. See Buja et al. (1991) for a mathematical treatment of this "null" case.

Larger values of p (with 3.0 being the highest setting on the scrollbar) have an opposite effect. Differences in dissimilarity are accentuated resulting in more "spiky" pictures. Figure 4 demonstrates the effect of setting $p = 2.0$ for the 5×5 grid graph.

Step size: The MDS update rule involves moving each object in the direction of the gradient of Equation 2.1. The *stepsize* scrollbar controls the size of the step in the direction of the gradient. Larger values cause the embedding to proceed rapidly, often skipping over dangerous local minima. However, they run the risk of causing the embedding to "thrash," that is, jump out of control from one configuration to another. Smaller values are useful when a figure has begun to settle.

Iteration control: With *n iters* set to 1, XGvis will update the graphics window after every MDS step. For more complex embeddings, it is sometimes more efficient to take several MDS steps before redrawing. Larger values of *n iters* accomplish this. The *do n iterations* button allows a user to perform a fixed number of MDS steps.

Distance metric: The user can bring up a menu of routines for recomputing the target dissimilarity matrix D. The metrics implemented include shortest path distance for graphs and Euclidean and Manhattan distances for multivariate data. This function is primarily useful for experimenting with different types of dimension reduction methods or in cases where there is ambiguity as to

what the default dissimilarity matrix should be.

3.2. Visualization with XGobi

When the optimization is running, the plot displayed in the XGobi window is redrawn with each iteration so that the motion of the points can be observed. If the data under study is a graph, setting the figure in motion while the optimization proceeds is particularly helpful. Using XGobi's *Rotate* or *Tour* mode, a viewer sees a smooth sequence of two-dimensional projections of the changing figure. Viewers describe this as watching the figure "unfold" or "puff out."

Once the user decides that the figure or point cloud has become stable, many of XGobi's other interactive functions can be used to advantage. Selecting the *Scale* mode allows the figure to be stretched or reshaped on the screen using mouse motions: this is especially useful if the viewer wants to zoom in on a subsection of a complex figure. In *Brush* mode, a user can interactively change the color of selected lines or points or change the shape and size of the plotting character used for selected points. Using *Identify*, a user moves the cursor near a point of interest, and its label appears. The label can be any user-supplied text string; by default, it is simply an identification number, 1 to n (the number of data objects). Point brushing and identification are linked: that is, if a user is working on the same data in another XGobi window (which could be part of another XGvis session or an independent instance of XGobi), then a point that is brushed or identified in one window is simultaneously affected in the second.

Some useful ways of selecting subsets by brushing and rerunning MDS on the brushed subsets were provided in an earlier implementation (Buja, 1982). These have not been included in the current XGvis system.

XGvis calls XGobi as a function (Swayne et al., 1991). In this use of XGobi, its data structures are defined by XGvis, the parent program, and yet all the interactive methods of XGobi are available for use on that data.

Figures 3 – 12 in this paper were produced using an XGobi facility which enables the user to create a Postscript^{TM}representation of the contents of the plotting window.

4. External Interfaces: NETPAD

Command line arguments allow a user to call XGvis in several ways: with a dissimilarity matrix, point locations, a graph specified as edges and point locations or a graph specified in NETPAD format.

Postscript is a trademark of Adobe Systems, Inc.

NETPAD is an interactive computer program for creating and analyzing graphs and graph algorithms. It contains a graphical user interface and an expandable toolkit of graph algorithms. It can be used like an electronic pencil and notepad to create, modify, save, recall and delete graphs and their associated attributes or attribute values. NETPAD has a library of programs for manipulating and analyzing graphs. These programs can be predefined functions which are part of NETPAD or user-defined programs such as XGvis which are interfaced to NETPAD.

All of the graph layouts displayed in NETPAD windows are two dimensional, and it is this limitation that XGvis enables it to surmount. Users can create and manipulate graphs using NETPAD and smoothly pass them to XGvis to be embedded in a higher dimensional space and then viewed using interactive motion graphics.

5. Examples

In this section, we provide two in-depth examples of applications of the XGvis system. They were chosen to demonstrate the range of possible uses for the system, from abstract mathematics to data analysis. The first involves using XGvis to construct and view a three dimensional layout of vertices for a well-known graph. In the second example, we use XGvis to find structure in empirical confusion data.

5.1. Adjacent Transposition Graph

Creating views of graph structures can be helpful to discrete mathematicians who are generating conjectures about attributes of a mathematical system. Graphs are useful in statistics for representing relationships between data measurements (Thompson, 1991). A good picture of a graph can make important trends in the data more apparent.

Creating higher dimensional views of graphs was one of the primary motivations for embarking on the XGvis project. We identified numerous methods for moving from a set of edges and vertices to a set of spatial locations for those vertices. We implemented and studied several of these including spring embeddings, isometric decomposition (Winkler, 1987), and a higher dimensional generalization of the Tutte embedding (Tutte, 1963).

Most of our efforts were focused on the multidimensional scaling algorithm described in this paper. Our embedding strategy was to make an analogy between shortest path distance in the graph and Euclidean distance in the resulting embedding. Starting with a graph represented as a list of vertices and edges, we compute,

using dynamic programming, a matrix D in which $D_{i,j}$ is the number of edges along the shortest path between vertex i and vertex j in the original graph.

We start with some initial assignment of vertices in the graph to positions in k dimensional Euclidean space; this initial mapping could simply be random. Then we use iterative multidimensional scaling to move the vertices to positions that better reflect the shortest path distances in D. Larger k's make it possible for MDS to fit the data better while smaller k's give humans a better chance of interpreting the resulting layout. The most instructive pictures involve trading off these forces and choosing the smallest k that appears to capture the significant structure in D. Often this involves a bit of trial and error.

Let us consider a graph, shown in Figure 5, known as the $n = 4$ adjacent transposition graph. This graph has special meaning to discrete mathematicians. It was passed on to us by Winkler, who was using it to study extensions of partial orderings. Thompson (1991) has also used this graph to depict relationships between surveyed preference judgments. For exploratory and explanatory purposes, it is useful to have a drawing that shows nodes as close together if and only if they are close together in the actual graph.

The graph is generated as follows. We start with all permutations of the sequence 1,2,3,4. There are 4! or 24 such sequences and we make each a vertex in the graph. We connect two vertices by an edge if one permutation can be turned into the other by simply transposing two adjacent elements. Figure 5 shows an initial layout for this graph; three of its nodes are labeled with their corresponding permutations. Observe that the node corresponding to 1423 is connected to 1243 because one can be turned into the other by swapping the middle two digits. However, 2134 is not connected to 1423 because there is no way to move from one to the other with a single adjacent flip. How many flips would it take? What we'd like is a representation of the graph that makes that easier to see.

Figures 6, 7 and 8 show XGvis finding the optimal three dimensional Euclidean layout for this graph, with the power of D set to its default value of 1.0. Figure 6 shows an intermediate stage in the "unfolding" of this graph, and Figures 7 and 8 show two views of the final position determined by the MDS algorithm.

A three dimensional embedding clarifies the structure a great deal. The resulting shape is equivalent to a truncated octahedron. One striking feature of the graph is its collection of six cycles and four cycles – hexagons and squares. Each six cycle is the collection of all permutations in which one end is held fixed and all permutations of the remaining three digits enumerated. Each square consists of the permutations which have the same symbols in the first two positions and the last two positions.

What was invisible is now clear. We can see in Figures 7 and 8 that 2134 and 1423 are just 3 flips apart and that there are two paths of that length. Interestingly, the sequence 1243 lies on both of them. The pair of nodes corresponding to 2134 and 1243 are both on a single square face. This face consists of all sequences which start with 1 and 2 and end with 3 and 4. The pair 1243 and 1423 are together on 2 hexagonal faces, one represents all sequences ending with 3 and the other all those starting with 1.

We have used XGvis to examine other graphs from this family including $n = 3$ (3! = 6 nodes connected as a hexagon) and $n = 5$ (5! = 120 nodes which requires 4 spatial dimensions to draw sensibly and is therefore difficult to visualize).

5.2. Morse code confusion data

In the previous example, the dissimilarity matrix that XGvis derived was based on the structure of a graph. Now we experiment with fitting an explicit dissimilarity matrix.

In Figure 9, each point represents a Morse code sequence – one for each of the symbols a-z and 0-9. A matrix of *confusion rates* for these symbols was determined in a psychological experiment by Rothkopf (see Gnanadesikan, 1977, pg. 44ff for an accessible account). To prepare the data for XGvis, we needed to convert the confusion rates – the percentage of times one symbol was mistaken for another – to dissimilarities. This was accomplished by subtracting each of the confusion rates from 1. The resulting matrix is recognized by XGvis as asymmetric and is made symmetric by the transformation described in Section 2.1. It has been discovered empirically that a good layout is found for this data in two dimensions. This is especially true if the dissimilarity matrix is transformed by cubing each cell; i.e., let $f(D_{ij}) = D_{ij}^3$.

Starting from a random layout, as shown in Figure 9, the points gradually migrate to positions that better approximate the target dissimilarity matrix. With the user watching in anticipation, they soon sort themselves out. The final layout is shown in Figure 10. It has been augmented by hand with some information useful for interpreting the result.

Figure 11 contains an XGobi plot of the percentage of dots used to represent each symbol versus its length in characters. (Recall that symbols are represented in Morse code by a sequence of one to five dots and/or dashes.) The plot in Figure 11 has been brushed along

the horizontal axis so that symbols with the same length representation in Morse code are drawn with the same glyph; for example, symbols represented by two characters (specifically i, a, n and m) are plotted using an "x".

The XGobi plot of *% dots* versus *length* in Figure 11 is linked to the XGvis plot in Figure 10, so the points in Figure 10 are similarly brushed. One symbol of each length has been labeled: the letters t, m, o, j and the number 0. Note that in the MDS representation of the confusion data, points form groups based on the length of the corresponding Morse code sequence.

Within each clump, the symbols have arranged themselves so that Morse code symbols with a high fraction of dots are above and to the left while those with relatively more dashes are below and to the right. This is shown in Figure 11, which was created by once more brushing the XGobi plot, this time along the vertical axis. The one-character and two-character symbols are all labeled: e (one dot and thus 100% dots) and t (one dash, 0% dots) are now brushed differently; i (two dots, 100% dots) is brushed differently than a and n (dot-dash and dash-dot, 50% dots) and m (two dashes, 0% dots).

Information such as this can be of particular use to code designers. By understanding the patterns of confusion, it might be possible to generate a new code that is less prone to misinterpretation.

6. Conclusion and Extensions

XGvis provides, in a single program, modules for creating, embedding, and displaying dissimilarity data. It is remarkably general in scope, providing useful functionality to mathematicians, statisticians and psychologists. The integration of computation and display make using the system natural; useful work can be accomplished by someone who has no expert knowledge of multidimensional scaling.

We have planned to extend the XGvis system with several other embedding algorithms. Nonmetric MDS (described in Section 2.2) has been used successfully for embedding graphs (Fairchild et al., 1988). Extensions are possible for drawing directed graphs – for instance, constraining edges to point down and to the right if possible. It is also possible to add additional terms to Equation 2.1 to encourage MDS to lay out the objects within a certain rough outline. This can be important for graphical layout applications.

The XGvis system described in this paper is not publicly available at the time of this writing. However, the XGobi and NETPAD systems on which it is based are available. Contact the authors for details (dfs@bellcore.com for XGobi; nate@bellcore.com for NETPAD).

7. Acknowledgments

We'd like to thank Paul Tukey for taking the trouble to type in the Morse code confusion data.

References

Borg, I. and Lingoes, J. (1987). *Multidimensional similarity structure analysis.* Springer-Verlag.

Buja, A. (1982). MDS in an interactive environment. Half hour film, ORION project, Stanford Linear Accelerator Center.

Buja, A., Logan, B. F., Reeds, J. R., and Shepp, L. A. (1991). Inequalities and positive-definite functions arising from a problem in multidimensional scaling. Bellcore and AT&T Bell Laboratories Technical Memoranda.

Dean, N., Monma, C. L., and Mevenkamp, M. (1991). NETPAD User's Guide. Bellcore Technical Memorandum.

Fairchild, K. M., Portrock, S. E., and Furnas, G. W. (1988). SEMNET: Three-Dimensional Graphic Representations of Large Knowledge Bases. In Guindon, R., editor, *Cognitive Science and Its Applications for Human Computer Interaction*, pages 201–233. Lawrence Erlbaum, Hillsdale, New Jersey.

Gnanadesikan, R. (1977). *Methods for Statistical Data Analysis of Multivariate Observations.* John Wiley and Sons, New York.

Kruskal, J. B. (1964a). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27.

Kruskal, J. B. (1964b). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29:115–129.

Kruskal, J. B. and Wish, M. (1978). *Multidimensional Scaling.* Sage Publications, Beverly Hills.

Swayne, D. F., Buja, A., and Hubbell, N. (1991a). XGobi Meets S: Integrating Software for Data Analysis. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pages 430–434, Fairfax Station, VA. Interface Foundation of North America, Inc.

Swayne, D. F., Cook, D., and Buja, A. (1991b). User's Manual for XGobi, a Dynamic Graphics Program for Data Analysis Implemented in the X Window System (Version 2). Bellcore Technical Memorandum.

Swayne, D. F., Cook, D., and Buja, A. (1991c). XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S. In *American Statistical Association 1991 Proceedings of the Section on Statistical Graphics*, pages 1–8. American Statistical Association.

Thompson, G. L. (1991). Exploratory Graphical Techniques for Ranked Data. In *Proceedings of the 23rd Symposium on the Interface*, Fairfax Station, VA. Interface Foundation of North America, Inc.

Thompson, G. L. (1993). Generalized Permutation Polytopes and Exploratory Graphical Methods for Ranked Data. *Annals of Statistics*. To appear.

Torgerson, W. S. (1958). *Theory and Methods of Scaling*. John Wiley and Sons.

Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–767.

Winkler, P. (1987). The metric structure of graphs: theory and applications. In Whitehead, C., editor, *Surveys in Combinatorics*, pages 197–221. Cambridge University Press.

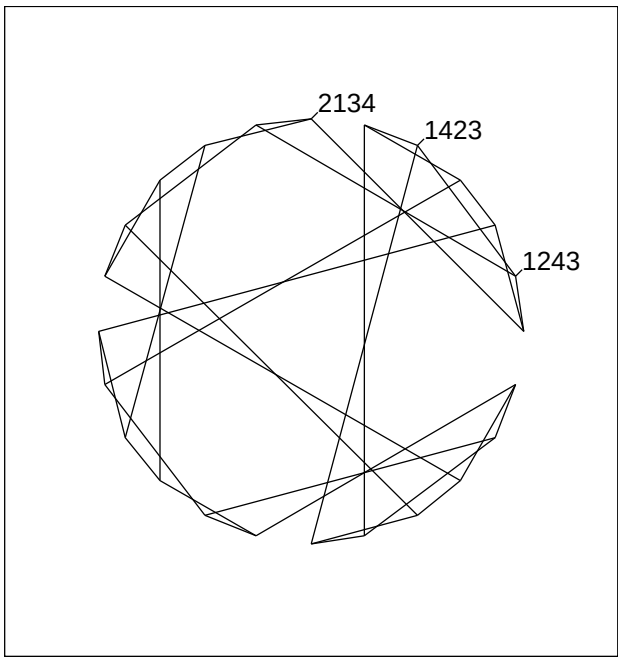

Figure 5: Adjacent transposition graph, first position

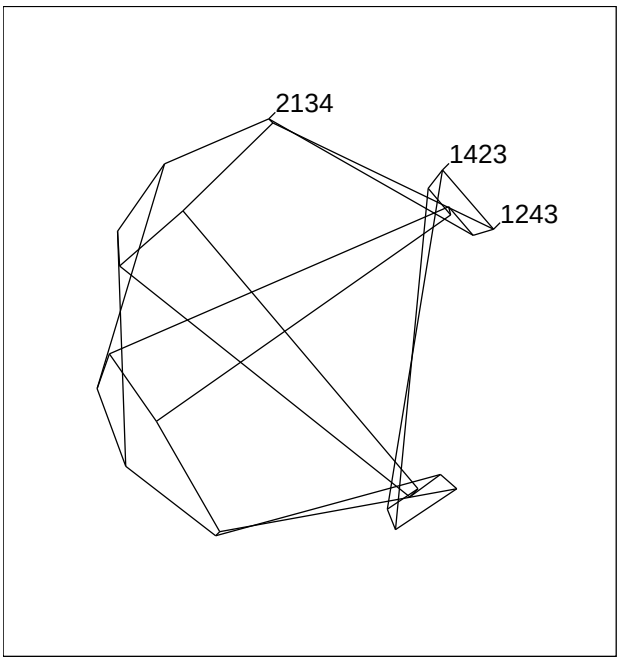

Figure 6: Adjacent transposition graph, unfolding

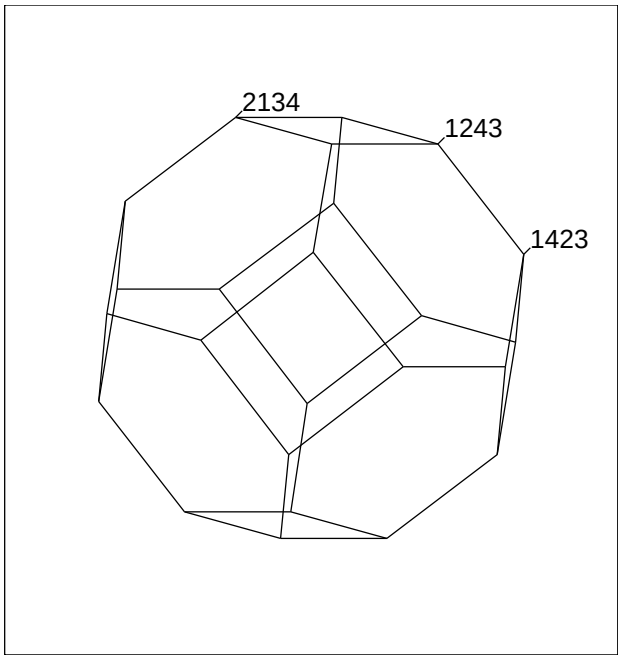

Figure 7: Adjacent transposition graph, final position

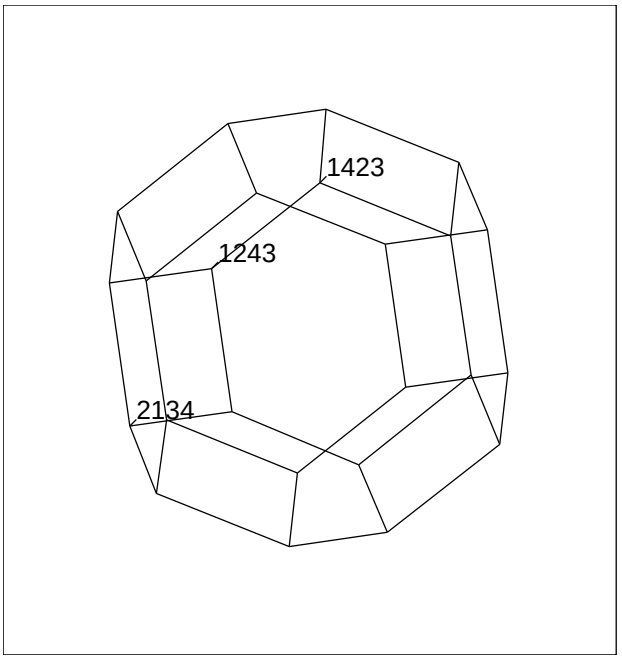

Figure 8: Adjacent transposition graph, rotated view

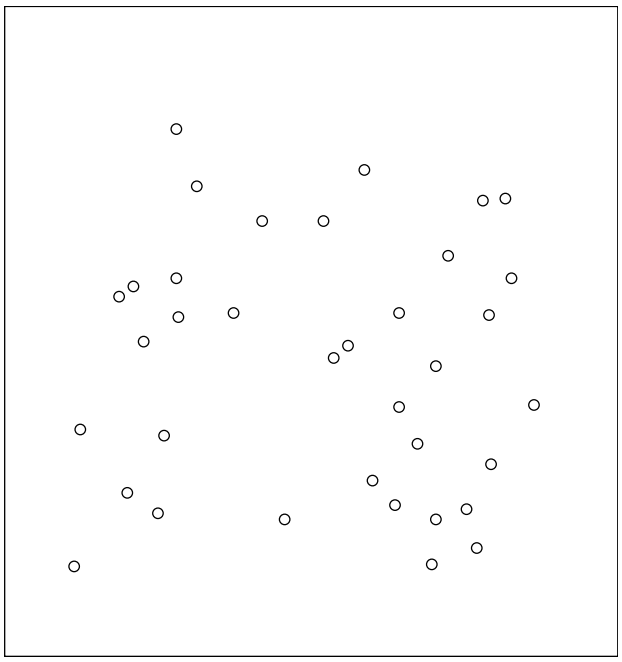

Figure 9: Morse code data, initial positions

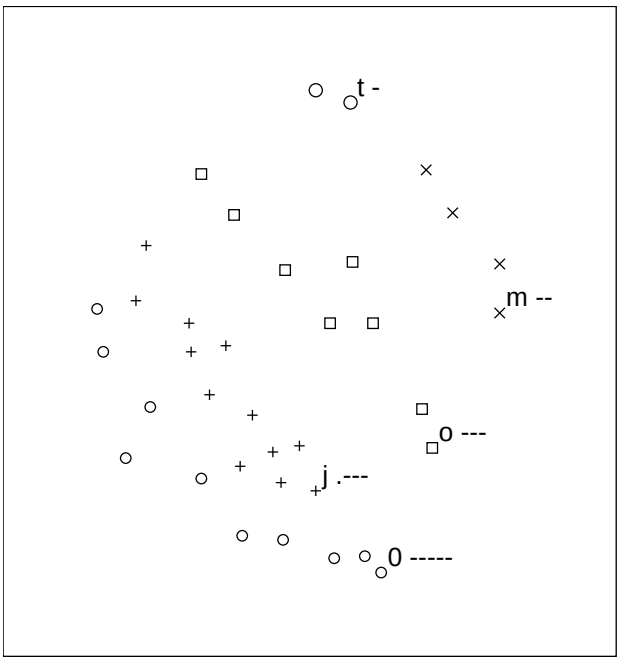

Figure 10: Morse code data, final positions; here the plot is brushed by the length of the Morse code representation of the symbol.

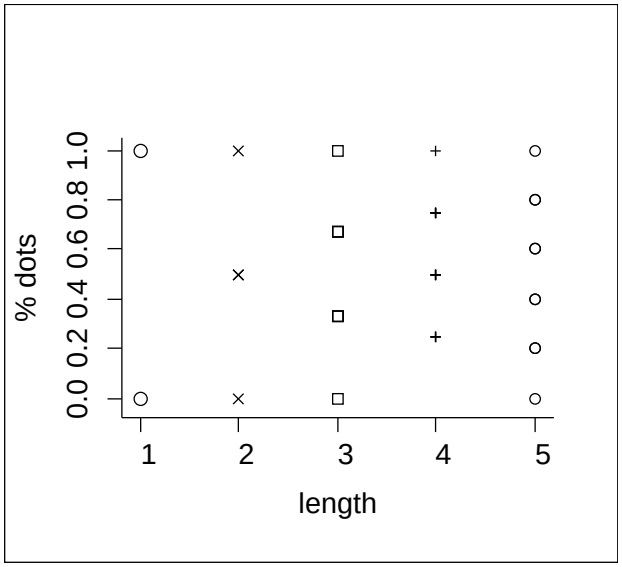

Figure 11: Morse code data, XGobi plotting window

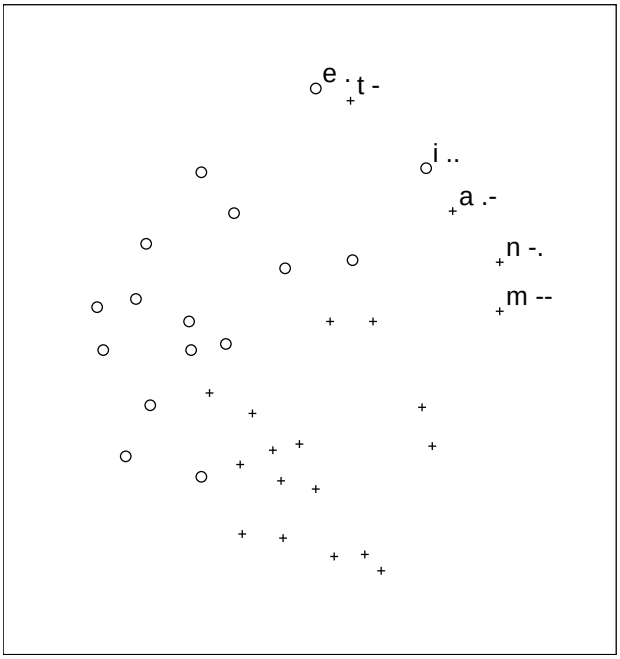

Figure 12: Morse code data, final positions; here the plot is brushed by the percentage of dots in the Morse code representation of the symbol.