

Mapping Mental Spaces:  
How We Organize Perceptual and Cognitive Information

by

Stephen A. B. Gilbert

B.S.E., Civil Engineering & Operations Research  
Princeton University, 1992

Submitted to the Department of Brain & Cognitive Sciences in Partial Fulfillment  
of the Requirements for the Degree of

Doctor of Philosophy in Cognitive Science

at the  
Massachusetts Institute of Technology

~~February 1997~~  
[June 1997]

© 1997 Massachusetts Institute of Technology  
All rights reserved

Signature of Author .....  
Department of Brain & Cognitive Sciences  
February 5, 1997

Certified by .....  
Whitman Richards  
Professor of Cognitive Science  
Thesis Supervisor

Accepted by .....  
Gerald E. Schneider  
Professor of Neuroscience  
Chairman, Department Graduate Committee

MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY

FEB 07 1997

ARCHIVED

LIBRARIES

# Mapping Mental Spaces: How We Organize Perceptual and Cognitive Information

by

Stephen A. B. Gilbert

Submitted to the Department of Brain & Cognitive Sciences  
on February 5, 1997 in Partial Fulfillment of the  
Requirements for the Degree of  
Doctor of Philosophy in Cognitive Science

## ABSTRACT

People use different representational forms to organize different kinds of knowledge. In the field of perceptual scaling, the two traditional techniques are multidimensional scaling (MDS), which uses a metric space as its representational form, and hierarchical clustering, which uses a tree. Both methods model subjects' similarity ratings of stimuli. We discuss a newer method, Trajectory Mapping (TM), that uses a connected graph as its representational form and models subjects' sequencing of stimuli. We compare TM with other scaling techniques (especially MDS and hierarchical clustering), at both a theoretical and data-driven level, and show that Trajectory Mapping models can offer insights that the traditional methods do not.

An algorithm for objectively generating trajectory maps from subject data is introduced and analyzed. Trajectory maps constructed by the algorithm are compared with previously published maps constructed by a manual heuristic and are seen to be satisfactorily equivalent, if not better in the case of particularly complex data sets.

Finally, a trajectory mapping experiment on the domain of mental representations themselves is described. Despite the usual division of representations into pictorial and linguistic categories, our results emphasize the importance of "mixed" representations, i.e. those with both pictorial and linguistic elements. The data also suggest that when subjects classify representations based on use, they do so using computational, functional, and graphical characteristics of the representations.

Thesis Supervisor: Whitman Richards  
Title: Professor of Cognitive Science

## Acknowledgments

I am grateful to Whitman Richards for being the shining lighthouse during my journey through the foggy waters of graduate school. His mentoring and inspiration were invaluable; he kept me afloat, and even when I had little idea of where the sought-after coastline lay, he confidently beckoned me onward.

To Josh Tenenbaum I owe enormous thanks for his ideas, as well as for his constant insistence on theoretical strength in my work and in our many discussions.

I thank Susan Carey, Steve Pinker, and Ken Haase for serving on my committee, for taking the time to read this document thoughtfully, and for their insights throughout my time in graduate school.

Jan Ellertsen deserves many accolades for giving me the strongest administrative support I can imagine, from directing me to my first Cambridge apartment to offering an ever-available ear to difficult personal issues to sustaining my yearnings to lead people.

I give thanks to all my friends in the department; they have laughed at my jokes despite themselves and tolerated my moments of stress. I thank especially the Gang of Five for starting my MIT life off right.

Progressing this far would have been impossible without my family and their constant encouragement. I thank Polly and Ray for the trust that all their educational investment, no matter how esoteric, would be useful someday, and I especially thank Frank and Libby for nurturing my faith during these years when it has been most needed.

I praise the saints of Mass. Ave. Baptist Church for their love and watch care while away from home these years. Most importantly, I sing Halleluia to God, the source of all ideas and maker of all things, for his most undeserved but continual blessings.

Funding for graduate work provided largely by a training grant from the National Institute of General Medical Sciences, #T32GM07484.

## **Biography**

After growing up frequenting the corners of libraries, the author entered Princeton University in 1988 and originally intended to be an architect so that he could play with an adult form of Legos forever. He was wooed from the architecture studio to the civil engineering department, however, by a cluster of Silicon Graphics workstations. While majoring in operations research and studying human-computer interaction, he decided that a common difficulty in solving real-world problems is not being able to grasp enough variables simultaneously. He came to MIT's Department of Brain & Cognitive Sciences in 1992 hoping that studying the mind and the brain might offer insights to this problem. Although he has found some, he continues to search.

## **Papers Related to this Dissertation**

Lokuge, I., Gilbert, S.A., & Richards, W. Structuring information with mental models: A tour of Boston. *Proceedings of ACM SIGCHI '96*, Vancouver, 413-419. See: [www-bcs.mit.edu/~stephen/chi96](http://www-bcs.mit.edu/~stephen/chi96)

Gilbert, S.A. & Richards, W. Using trajectory mapping to analyze musical intervals. *Proceedings of the Conference of the Cognitive Science Society, 1994*, Atlanta, 363-368. See: [www-bcs.mit.edu/~stephen/cog94](http://www-bcs.mit.edu/~stephen/cog94)

## Table of Contents

Abstract.....	2
Acknowledgments.....	3
Biography & Related Papers.....	4
Chapter 1.....	7
<b>Modeling Mental Representations</b>	
Chapter 2.....	29
<b>A Data-Driven Comparison of TM with MDS and Hierarchical Clustering</b>	
Chapter 3.....	45
<b>An Objective Approach to Trajectory Mapping through Simulated Annealing</b>	
Chapter 4.....	65
<b>Testing the Algorithm: Automatic vs. Manual Trajectory Maps</b>	
Chapter 5.....	81
<b>A “Meta” Problem: Representational Forms and Functions</b>	
Chapter 6.....	97
<b>Conclusion &amp; Summary</b>	
Appendix A.....	101
<b>Deeper Analysis of the TM Algorithm: Work in Progress</b>	
Appendix B.....	107
<b>Additional TM Data Sets</b>	
Appendix C.....	113
<b>Algorithm Code and Details</b>	



# Chapter 1

## Modeling Mental Representations

### Abstract

We compare several approaches to cognitive modeling and their representational forms, including multi-dimensional scaling (metric space), hierarchical clustering (clusters), and Trajectory Mapping (connected graph). Trajectory Mapping is explicitly contrasted with other approaches, especially those that also use connected graphs.

### Part I: Introduction

#### The Two Questions

People use different mental representations to organize different kinds of knowledge. Our knowledge about the characteristics of various vegetables, for example, is probably structured differently than our knowledge of a fairy tale that we learned in childhood. Also, different people often use different mental representations to organize the same knowledge. We are often vexed by the different ways that other people structure spatial navigation information; some people do not like maps, some people use north, south, east, and west but not left and right, some people prefer pictorial representations, and some prefer linguistic representations.

Two questions that follow naturally from these differences are:

- 1) What is the structure of someone's mental representation of given domain?  
and
- 2) Which characteristics of the domain are used to organize the mental representation?

This thesis addresses these questions by exploring a relatively new method of modeling mental representations, called Trajectory Mapping (TM), in the context of two other common approaches. We suggest that TM fills an important role in the field because it uses a connected graph as its structural representation and because, unlike many other approaches to cognitive modeling, it does not explicitly use similarity judgments as the data for its model. We offer more details of our specific approach below; first, we define our terms and motivate the two stated questions.

By the "structure" of a mental representation we mean a computational structure, a structure that can be described mathematically and implemented with a computer. We explicitly do *not* mean a neurological structure, although the computational structures that we seek could be implemented in neural hardware as well as computer hardware. By "domain" we mean a

conceptual category, as described by work in the area of conceptual structure (e.g. Smith, 1989). Examples of domains are "fish", "inanimate things", "colors", "melodies", etc. Lastly, when we ask what characteristics are used to "organize" the mental representation, we ask which features of the exemplars in the domain are used as the principal axes of or the basis for arranging the exemplars within the structure. For example, we might say that the domain of colors can be modeled as a three-dimensional coordinate space; "3-D metric space" would be the computational structure. If the dimensions of that space are hue, saturation, and brightness, then we say that those features are used to organize the space.

Being able to answer these two questions reliably for any domain would be enormously helpful not only for the field of cognitive science, but also for a variety of applications. The line of research that aspires to answer these questions is broadly called cognitive modeling. Cognitive modelers seek to model various aspects of human knowledge so that we can both understand better how the mind processes information and make accurate predictions about human behavior. The range of the field can be well illustrated by the list of different types of mental representations that have been proposed: cognitive maps (Tolman, 1948), semantic nets (Quillian, 1968), production systems (Newell, 1973), frames (Minsky, 1975), scripts (Schank & Abelson, 1977), schemata (Rumelhart & Ortony, 1977), and mental models (Gentner & Stevens, 1983; Johnson-Laird, 1983). In the case of our two questions specifically, we could learn easily what features people consider most salient in a novel domain of knowledge.

The applications of cognitive modeling are numerous. An accurate model of human knowledge organization could give us inspiration for building better computerized processing of knowledge. Studying how we index our mental representations might give us insight into indexing the plethora of knowledge now available online. Good modeling of knowledge could provide educators with more details about learning differences and guide curriculum design. Marketers, of course, are also deeply interested in how we organize our knowledge about their products.

### **The Direct vs. Indirect Approach**

There are several potential approaches to answering the two questions above. The first that might come to mind is the direct approach: to discover what sorts of mental representations and features people use, simply ask them. One might imagine experiments in which we present a subject with a collection of 20 vegetables and say, "Arrange these into some sensible organization that seems natural to you" (question 1). After they have done so, we ask, "What features or principles did you use to organize them?" (question 2). This approach suffers from two basic problems. The first is that our success in eliciting features depends strongly on the ability of subjects to introspect about what they did and on their ability to verbalize their introspection. The large range of these abilities in the typical subject population suggests that subjects' tasks should be simpler and more consistent across subjects.



The second problem with the direct approach is that any sort of grouping or clustering task is subject to getting caught in "local minima" as a result of the initial steps. If a subject initially groups together all the green vegetables, for example, he might not realize that the green pepper could also be put with the jalapeño and the chili pepper. This example also points out the difficulty of assigning stimuli to multiple groups (overlapping clusters) when grouping by hand. These difficulties suggest an approach that asks subjects to interact with only a few stimuli at a time, making judgments about "local" structure within the stimulus set. The experimenter would then hopefully use this information to make conclusions about the "global" structure within the stimuli.

Having explored the difficulties with a direct approach to modeling mental representations, we describe an indirect approach that tries to alleviate these difficulties. Contrasting and comparing versions of this approach will be one goal of this thesis. The indirect approach, classically used in the research area of psychophysics, typically involves the use of scaling techniques that use subjects' judgments to construct a model of the global structure of the stimuli. To use a scaling technique, we might ask a subject to give numerical ratings for the similarity of two stimuli or ask questions such as, "Of these three stimuli, which two are more closely related?" Given such data, a variety of scaling techniques are available; each provides an algorithm which constructs a computational model of these ratings, such as a Cartesian coordinate space, or a network of nodes and links. The experimenter then searches the model (based only on local judgments) for new global information about the stimuli. This information might take the form of obvious axes in a coordinate space or obvious clusters or chains in a network. By applying prior knowledge about the domain, the experimenter can sometimes denote these parts of the model as representing the most salient features of the domain.

The careful reader might protest that this approach is circular, since the experimenter uses prior knowledge of the domain to label the "new" knowledge of the domain. A simple example reveals that this is not the case. In the domain of colors, the experimenter's prior knowledge might consist of knowing the frequency spectrum and that colors can be modeled by a three-dimensional space with axes of red, green, and blue. These parameters may have little to do with a subject's mental representation of color, however. The subject might have a mental representation based on hue and a mixture of saturation and brightness, for example, or a representation that simply groups colors into overlapping groups by general hue, and then has a separate group of grays. If the scaling technique offers a good model of such representations, the experimenter's prior knowledge would be useful for figuring out how the model was organized, but it would not be synonymous with the knowledge inferred from the organization.

If this indirect approach succeeds, then we have found a reasonable method for answering our two questions while overcoming the difficulties mentioned in the direct approach: subjects make only simple judgments about only local stimulus relations. The next issue that must be answered is how we can know

whether we succeed. The issue of success in modeling mental representations is particularly difficult because knowledge of a given domain could easily take several forms simultaneously. Nevertheless, there are two basic methods of measuring the success of this approach. The first is find independent confirmation of the results through a variety of other experimental techniques. If a variety of techniques that do not use the same assumptions arrive at similar answers, then the results can be considered successful. Secondly, if the resulting model can be used to predict behavior accurately, especially about non-tested domains, then the model can be considered successful as well.

Different approaches to cognitive modeling differ mainly in their choice of computational structure and the type of data they solicit from subjects, or in other words, what sort of data they are represent with what structure. Table 1 offers an overview of various approaches and some of their differences. The two most commonly used methods in the field (shaded) are multidimensional scaling (MDS) and hierarchical clustering.

Once again, the goals of this thesis are to explore how well Trajectory Mapping can help us answer the two questions, and to place it, as an approach, into the context of the field and the other methods. As one can see in Table 1, a salient contribution of TM is that it offers an approach based on subject's ordering of stimuli instead of similarity judgments. The use of orderings makes intuitive sense, since many mental representations that we would like to model may be based on properties that specifically link objects together, such as "is smaller than" or "contains". Also, the computational structure assumed by TM differs from those of MDS or clustering. A detailed comparison of methods and their computational structures follows.

Method	Computational Structure	Type of Subject Data
<b>Multidimensional Scaling</b>	<b>metric space</b>	similarity judgments
<b>hierarchical clustering</b>	<b>nested clusters</b>	similarity judgments
Trajectory Mapping	connected graph	orderings
Pathfinder	connected graph	similarity judgments
NETSCAL	connected graph	similarity judgments
MAPNET	connected graph	similarity judgments
Concept Mapping	connected graph	conceptual associations
Repertory Grid Theory	list of constructs/features	scalar judgments by feature

**Table 1:** Various approaches to modeling mental representations, the different computational structures that they assume, and the type of data they demand from subjects. Most frequently used methods are shaded.

## Part II: Introduction to the Top Two Methods and TM

Because we wish to contrast the computational structures used in each approach, it is appropriate that the reader understand the difference between the methods. In this part we describe two of the most traditional methods for modeling mental representations of stimuli, MDS and hierarchical clustering. We then introduce Trajectory Mapping as a technique and describe the connected graph that it uses as a representation.

### Multi-Dimensional Scaling (MDS)

Multi-Dimensional Scaling, first discussed by Torgerson (1952), Shepard (1962), and Kruskal (1964), places stimuli in a metric space based on a matrix of proximities. The proximities matrix contains either similarities or dissimilarities between all the stimuli, and the output is the spatial arrangement of the stimuli that maximizes the fit between the distances in the metric space and the proximities. Usually a non-linear function provides the mapping between the two sets of measurements.

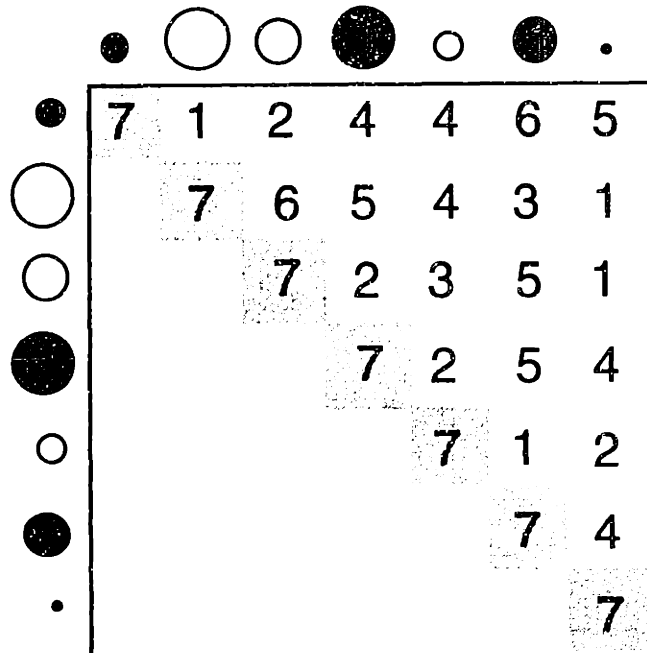
Several noteworthy assumptions must hold in MDS. The proximities matrix must be symmetric, that is, the distance from A to B is the same as from B to A. Also, it must be reasonable to assume that the stimuli could lie in a space in which a uniform distance metric holds (typically Euclidean). The only parameter in MDS is the dimensionality of the final metric space. The algorithm provides a measure-of-fit diagnostic called the "stress", which can be used to help choose the dimensionality. Because the stress inevitably decreases with higher dimensions, the experimenter typically chooses the dimensionality at the last appreciative decrease in stress.

Figure 1 shows an example of a typical similarity matrix for a simple domain of black and white circles of different sizes, an example domain introduced by Richards & Koenderink (1995). For this domain, one would show subjects each possible pair of circles and ask, "On a scale from 1 to 7, how similar are these two circles to each other?" The maximum similarity, 7, is shown shaded along the diagonal. One can then use algorithms such as KYST2 (Kruskal, 1976) to run MDS and produce the coordinates for the stimuli. Figure 2 shows the result of MDS on our black and white circles example. The KYST2 algorithm assigns coordinates to the circles so that the distances between them in the space most closely reflects the similarity ratings (e.g. high similarity = small distance).

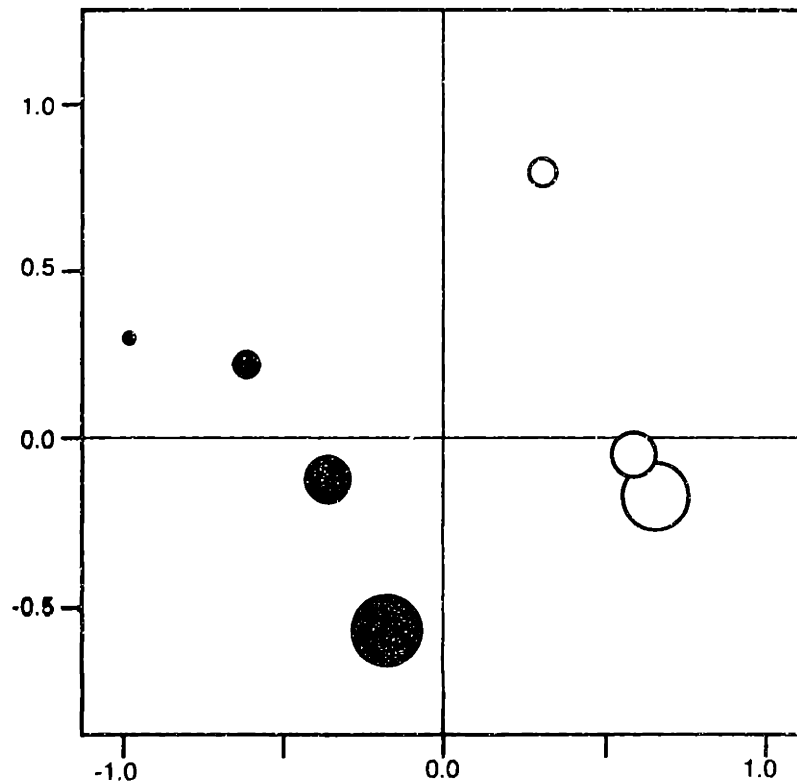
Because MDS offers a representation of the stimuli with a lower dimensionality than the original representation, researchers hope that MDS will produce a space that might help one infer the primary features of the subjects' conceptual model of the stimuli. In Figure 2 one can see that MDS has generally captured the two features of the circles, size and color: the white circles are further right, the black ones are further left, the smaller ones are higher, and the larger ones are lower. Ideally, MDS would position the circles such that the principal

components of the space reflect these two features exactly; here, we are close, but the axes are a little bit skewed.

We remind the reader that while it may seem useless to model a domain with two obvious features and infer those same features from the model, we use this domain to illustrate the techniques. Much work in this field is done initially with simple artificial domains to test the robustness of the techniques. Once proven, the techniques can then be used to explore novel or very complicated domains.



**Figure 1:** An example of a similarity matrix of the sort that could be used for metric scaling or hierarchical clustering. Circles with a higher similarity have higher similarity values.



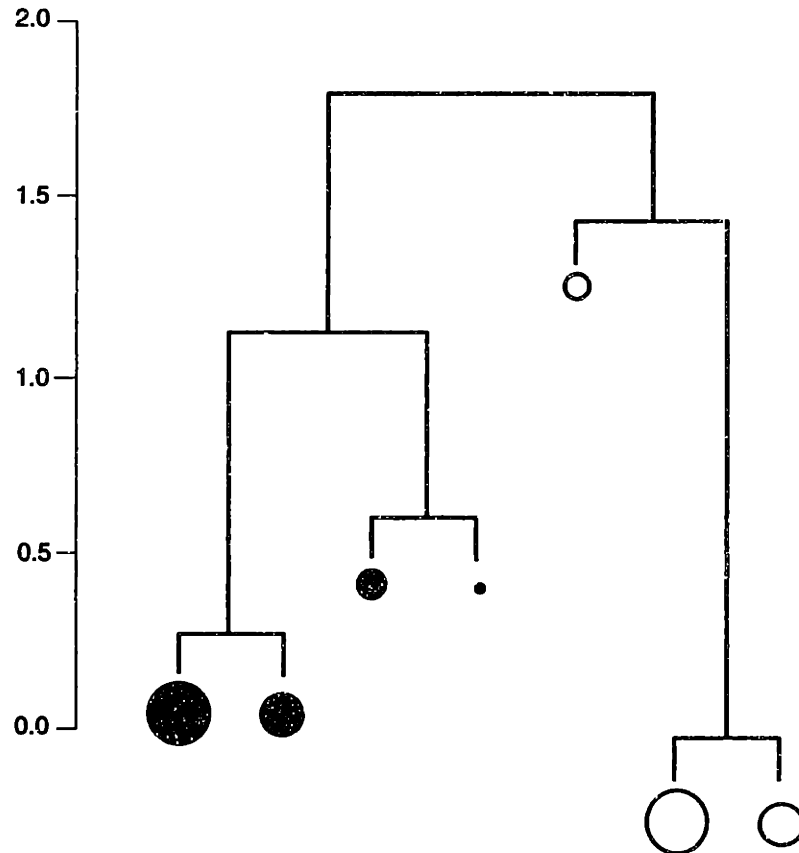
**Figure 2:** The output of Multi-Dimensional Scaling (MDS) on the similarity values in Figure 1, stress = 0.15. The axes generally reflect the two features of the circles, their color and size.

### **Hierarchical Clustering**

Subjective visual inspection of Figure 2 suggests two clusters. Hierarchical clustering, another classic scaling technique, makes this clustering explicit in the form of a dendrogram or tree. This technique is much mathematically simpler than MDS, but also has few formal mathematical properties. Typically one begins clustering by considering each data point to be a cluster in itself and then merges the clusters one-by-one iteratively until there is one cluster that encompasses the entire data set. This form of bottom-up clustering is called “agglomerative,” as opposed to “divisive” clustering, where one begins with the entire data set and iteratively divides. The main parameter that must be chosen with hierarchical clustering is the distance function that one will use to determine which two clusters are closest at each iteration.

Depending on what sort of distance function is chosen, a variety of different cluster partitions can result from the same data. Duda & Hart (1973) and Corter (1996) offer a good explanation of the differences. Briefly, in single linkage or connected trees, the distance is the minimum distance between two points in the two clusters in question. Use of this distance function is also called the nearest-neighbor algorithm. In complete linkage or compact trees, the distance is the maximum distance between two points in the two clusters in question. This method is also called the furthest-neighbor algorithm. Lastly, some researchers use an “average” distance function which is based on the

average of the distances between points in the clusters. Most hierarchical clustering in this paper use complete linkage, though the tree in our black and white circles example (Figure 3) uses average linkage to better illustrate our point.



**Figure 3:** The output of hierarchical clustering (average linkage) on the similarity values in Figure 1. The top-most cluster reflects the color difference; the continuous feature of size is not clearly illustrated by this discrete representation.

It is worth noting the debate around the choice of representational form in the history of this field. Shepard proposed his technique of non-metric MDS on similarities in 1962, but later described six problems with finding the structure in any similarity-based data (1974). Tversky published an alternate view of similarity based on set-theory (1977), adding weight to the problems posed by Shepard. Tversky pointed out that similarity data often do not fulfill the basic assumptions of metric theory, i.e. minimality, symmetry, and the triangle inequality. He cited his classic example of asymmetry in similarity judgments: North Korea is more similar to China than China is to North Korea. He proposed that instead of embedding stimuli in a metric space, one could describe them as members of categorical sets based on their common and distinctive features. He favored clustering trees as a representation for such stimuli.

Pruzansky, Tversky, and Carroll (1982) did a large comparison of representations over 20 data sets, modeling each with both clustering and MDS and comparing the models using several objective measures, such as the skewness of the distribution of distances. They suggested that although similarity data can be modeled with both MDS and clustering, particular data sets are modeled better by one method or the other. Most notably, the fit of the models depended on both the type of domain, e.g. perceptual vs. conceptual, and on the method used for sampling the domain to choose the stimuli. MDS was better for perceptual stimuli and stimuli chosen in a factorial design, while clustering seemed more appropriate for conceptual stimuli, such as semantic categories, that had been chosen by "natural selection" (p. 18).

TM arises generally from this history. Originally motivated by the problems with similarity described by both Shepard and Tversky, TM attempts to offer a representation that, like metric spaces and clustering trees, allows researchers to infer new information about the stimulus set and the features used by a subject to organize the stimuli mentally. TM diverges from these approaches, however, in that it does not build its representation directly from similarity judgments. Because similarity judgments are pairwise data, we suggest that traditional metric spaces and trees can fail to model serial aspects of a data set, i.e. ordered relations across multiple stimuli. Lastly, before continuing to the details of TM, we suggest that its representational form, a connected graph with weighted links, combines advantageous aspects from both MDS and clustering. Like in a hierarchical tree of clusters, the nodes in a trajectory map are typically clustered into subgraphs, and similar to the ordering of stimuli in an MDS metric space along dimensions, nodes are rank ordered within their subgraph.

### **Trajectory Mapping (TM)**

There are 3 stages to TM. The first is the experimental paradigm, i.e. collecting the data. The second is analyzing the data, i.e. turning it into graphs. The third is interpreting the data, i.e. deciding what the trajectories in the graphs imply about the subject's mental representations.

In the TM experimental method a subject initially surveys the entire range of stimuli. The subject's task in TM is to imagine a conceptual feature or property that links a given pair of stimuli. The subject then extrapolates that feature in both directions to pick two stimuli from the remaining set that would be appropriate. The subject also picks an interpolant, i.e. a stimulus that would fit well within the pair. We ask that the subject mentally use the same feature in each choice.

To clarify the procedure, we offer again the example of seven black and white circles of different sizes. The subject is first presented with the entire collection. In each TM trial, a pair of circles would be chosen and designated **A** and **B** (Figure 4). The subject is asked to note some feature that differs across the two and to choose from the remaining stimuli a sample for each of the blank spaces, an extrapolant in each direction and an interpolant.

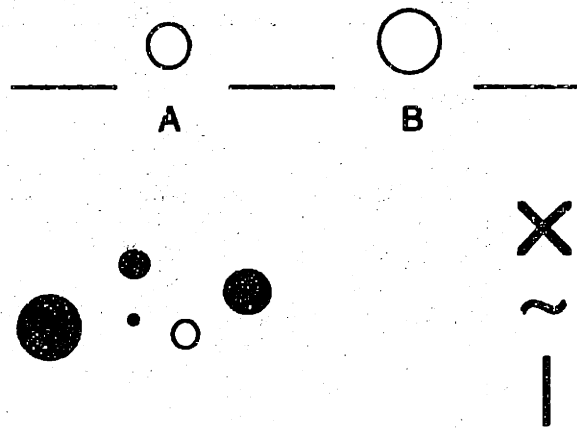


Figure 4: A typical TM trial; the subject chooses stimuli that fit in the blanks to make a good quintuplet sequence. The subject also has the option to choose the "X", meaning that she considers it "not feasible" to make a sequence from the pair. The "~" means that the subject can imagine filling the slot with an appropriate stimulus, but it is "not available" in the set. The "|" means that the subject has reached a "dead end" in the sequence, for example, if the sequence were based on size and there were no smaller circles.

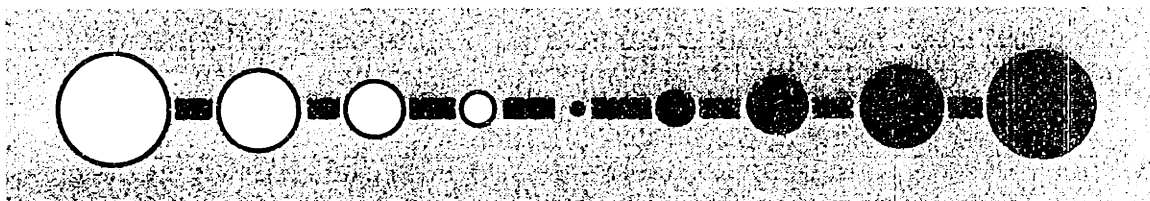
ex	A	int	B	ex	ex	A	int	B	ex
~	○	○	○	•	●	●	●	•	
X	○	~	●	X	~	○	~	○	○
•	●	~	●	●	X	●	X	○	X
	•	○	○	○	~	●	●	•	
~	●	~	●	●	○	○	~	•	
X	•	X	○	X	○	○	~	○	•

Figure 5: Example TM trials from the circles data set. All possible pairs of circles are presented as an A and B pair, and subjects are asked for extrapolants and an interpolant that complete a sensible quintuplet sequence.



Figure 5 shows some of the other trials. Note in the table that three special cases arise in this paradigm, and the subject is allowed to mark them appropriately. The first occurs when **A** and **B** are so dissimilar that the subject feels uncomfortable choosing one particular feature that varies between the two. We call this the "infeasible" case, marked with an "x". The second case is the "feasible, no sample" case (marked with a "~"), in which the subject can easily imagine an appropriate extrapolant or interpolant, but it is not included in the sample set. The third case, the "dead end" (marked by a "|"), occurs when either **A** or **B** represent an extreme in the variability range of the chosen feature.

Using these quintuplets of subject data, we use an algorithm based on simulated annealing to generate a connected graph or trajectory map, such as the graph in Figure 6. The links in the graph reflect frequent connections found in the quintuplet data. By further analyzing the trajectory map and the data that created it, we can note subgraphs that might represent features in the data. The hatching within the links shows two such subgraphs or trajectories; each trajectory represents a categorical feature (e.g. black circles) and each trajectory itself is ordered by a continuous feature (size). These subgraphs, although not always found in trajectory maps, offer a representation that combines features of both metric scaling and clustering; the subgraphs are ordered clusters.



**Figure 6:** The connected graph above is a trajectory map of the black and white circles. The subgraphs, identified by hatching within the links, illustrate the salient features of the stimuli, can be identified by thresholding and analyzing how the subject data fit along the trajectory map.

To give additional flavor for the types of features that a subject can use during TM, we offer several examples of three-element trajectories that one might find in a TM experiment. Note that some elements occupy in the same basic-level category, and the trajectory is formed by a simple feature within the category, while other trajectories are based on more abstract relationships, such as *contains*, *generates*, *rules over*, etc..

- (big, medium, small)
- (cool, tepid, warm)
- (rat, cat, dog)
- (house, living room, human body)
- (professor, graduate student, undergraduate)
- (grass, antelope, lion)
- (park, zoo, jungle)

Likewise, we offer some examples of pairs that would likely be deemed “not feasible”. There are three basic reasons why a subject would indicate “not feasible”: 1) the two stimuli are not in the same category or domain, and therefore the subject can find any relationship between them (jigsaw, elephant), 2) the stimuli are in the same domain and do have a relationship, but it is not the sort of relationship that can be used to form the basis of a sequence (computer monitor, computer), and 3) the stimuli are in the same domain, but they are too “far apart” in feature space for the subject to imagine a relationship (in the domain of background noises: traffic jam, wind).

(jigsaw, elephant)  
(kumquat, mutual fund)  
(computer monitor, computer)  
(traffic sounds, wind sounds)  
(Museum of Fine Arts, shoe shop)

### **Summary of Part II**

The methods of Multi-Dimensional Scaling (MDS), hierarchical clustering, and Trajectory Mapping (TM) have been introduced. Each method can be distinguished well by noted what representational form the method uses to represent what sort of subject data. MDS represents similarity judgments in terms of a metric space. Hierarchical clustering represents similarity judgments in terms of hierarchical clusters, often represented with a tree or dendrogram. TM represents orderings with a connected graph, thus distinguishing itself from MDS and hierarchical clustering in two ways. We noted briefly how a trajectory map can offer some of the advantages of each of the other methods: subgraphs of the trajectory map often represent clusters of stimuli (as in hierarchical clustering) that have been ordered by some feature (as in MDS).

### **Part III: TM vs. Potentially Similar Methods**

In this part we compare TM with the other methods listed in Table 1 at the beginning of the chapter. These methods should be carefully distinguished from TM because they share some characteristic. Most of them are similar in that they use a connected graph as their representational form. Repertory Grid Theory is somewhat similar to TM in the form of data it collects from subjects; both methods ask subjects to make judgments that require them to consider specific features of the stimuli.

#### **Pathfinder, NETSCAL, & MAPNET**

Researchers have long pursued graphs as representation for similarity (Harary, 1964) or for knowledge representation (Augustson & Minker, 1970; Collins & Loftus, 1975). Several more recent efforts work with graphs that are similar

enough in their purposes to TM that we should examine the differences between the various methods. The other relevant graph-based scaling techniques are called Pathfinder (Cooke, Durso, & Schvaneveldt, 1986; Cooke, 1992), NETSCAL (Hutchinson, 1989), and MAPNET (Klauer, 1989; Klauer & Carroll, 1989, 1991).

The Pathfinder algorithm builds a connected graph from a similarity matrix in such a way that the minimum-length paths between nodes  $i$  and  $j$  in the graph are inversely proportional to their similarities. (This goal is similar to the goal of MDS; instead of metric distances, Pathfinder uses graph distance.) Arriving at this solution means choosing both which links the graph will contain and what the weights should be on those links. Various powers of the Mikowski distance metric are tried as the mapping function between the similarities and the graph distances. The power,  $r$ , of the distance metric serves as one of the parameters for Pathfinder, and the other parameter,  $q$ , is related to the maximum number of links permitted in a path between nodes.

Because it is not obvious what measure one might use to compare whether a Pathfinder graph or an MDS metric space offers a better fit to given similarity data, Cooke, Durso, & Schvaneveldt (1986) and Cooke (1992) compare Pathfinder with MDS by testing which algorithm produces a model that better predicts reaction times in semantically related tasks. They find that Pathfinder models predict behavior better than MDS models in tasks demanding serial and free recall of lists, a task asking whether two stimuli share category membership, and a task asking the relation of two stimuli along a given dimension, i.e. greater or lesser. This performance illustrates that Pathfinder offers a good model of the features that subjects find useful in those tasks, but it does not tell us about the features that subjects might use to organize the stimuli in their mind. The authors suggest that Pathfinder usually represents "local relations" while MDS represents "global relations". We suggest that TM can span this range by clustering based on global relations and ordering the stimuli within the clusters according to local relations.

Hutchinson (1989) introduced NETSCAL as a solution to modeling asymmetric proximity data. Like Pathfinder, NETSCAL constructs a graph such that the minimum path lengths between nodes approximate the dissimilarities as closely as possible. NETSCAL has two parameters also, each part of the function that maps the proximities onto the minimum path lengths. Hutchinson demonstrates the robustness of the algorithm by fitting various Monte Carlo simulations of networks that have had noise added. Hutchinson highlights the asymmetric similarity modeling ability of NETSCAL with clothing and alcoholic drink datasets.

As Hutchinson points out, the NETSCAL graphs offer a better representation than MDS when many of the similarity values between pairs equal zero; MDS has no way of representing very many maximal distances in two or three dimensions. The NETSCAL graphs can also demonstrate some degree of hierarchy. In a stimulus set of birds that includes type of bird, e.g. *robin*, *owl* as well as the concept *bird*, one finds *bird* in the center of the graph where all

links meet. Similar to Pathfinder, however, these graphs have difficulty illustrating any large scale structure within the graph. If there were multiple domains in the stimulus set, with several levels of category and several exemplars at each level, it would soon become difficult to distinguish which nodes were superordinate and which were subordinate.

Also in 1989, Klauer and Carroll introduced the graph fitting algorithm that they would later call MAPNET (Klauer & Carroll, 1991). Like the other algorithms, MAPNET provides a graph which provides the best possible fit between the minimum path lengths between nodes and the similarity values. MAPNET is different, however, in that instead of determining the ideal number of links as part of the solution, MAPNET maximizes the goodness of fit for a user-specified number of links. Klauer and Carroll (1991) test MAPNET with some of the same Monte Carlo techniques and data sets as Hutchinson (1989), maintaining that MAPNET models account for more variance than NETSCAL in each data set. Overall, MAPNET produces similar graphs to NETSCAL, and therefore has the similar disadvantages in terms of finding feature-based structures within the graphs.

It is worth noting from the outset that the primary difference between TM and all of the other methods is that the others build graph-based models of similarity judgments, whereas TM builds a graph-based model of feature extrapolations, which can then be used to synthesize approximate similarity judgments. This difference manifests itself in the graphs themselves: while Trajectory Maps (TM graphs) often contain meaningful chains of nodes, meaningful linking in graphs from the other algorithms often appears distinctly pairwise. We suggest that because the other algorithms are indeed trying to model pairwise data, they do not take full advantage of the graph representational form to illustrate salient features of the nodes by showing any ordering within the graph. Also, because the similarity between two stimuli can encompass many features, while TM orderings refer usually to more specific features of the stimuli, similarity judgments are more prone to being skewed by the context of other stimuli in the data set.

### **Concept Mapping**

Concept Mapping is a technique described by Novak and Gowin (1984) for externalizing knowledge about a particular domain. The technique has been used mainly in the disciplines of education research, library science, and expert systems development and is very similar to the notion of a semantic net (Quillian, 1968). Basically, the technique involves a subject's deciding what the relevant concepts are in a given domain. The subject then orders these concepts according to their specificity, linking them in a hierarchical graph structure if possible. Finally, the subject assigns relevant meanings to the links. An example from Novak and Gowin (1984) is shown in Figure 7.

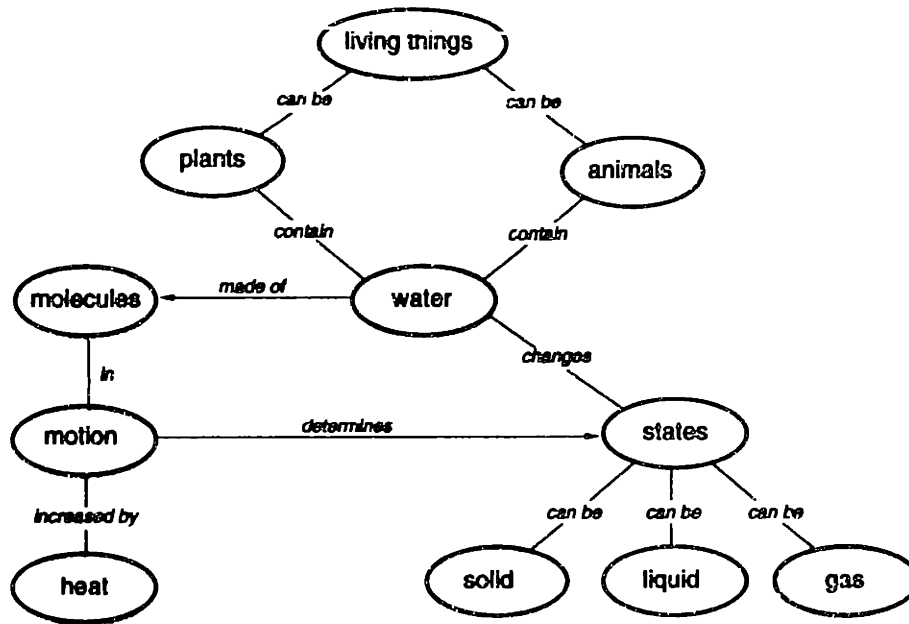


Figure 7: A concept map from Novak & Gowin (1984) for the knowledge associated with water.

We introduce concept mapping for the purpose of comparison to TM, since the output of the two techniques nominally resemble each other. Despite this resemblance, the two techniques differ in many ways. Firstly, concept mapping undertakes to represent a broader set of mental knowledge than TM. It aspires to represent the organizing principles of entire domains of knowledge, such as *basketball*, *library reference resources*, and *culture* (Novak & Gowin, 1984; Sherratt & Schlabach, 1990; Henderson, Patching & Putt, 1994). TM, on the other hand, attempts to infer the salient features used to organize exemplars of a single concept, e.g. Western musical intervals, colors, sound textures, musical intervals, or tourist attractions.

Secondly, concept mapping makes few claims about knowledge representation in general. It was originally designed for educational purposes, to help students structure their knowledge and allow teachers to see how they think about a domain. Much of the benefit of a subject's constructing a concept map is described as the subject's own gain in awareness of her knowledge. In this sense, concept mapping is more a tool to aid an individual's introspection about the knowledge domain than an analysis of the domain in general. It is worth noting, however, that the representational form of concept maps, i.e. concept nodes with association links, appears in a variety of AI approaches to knowledge representation, notably semantic networks (Quillian, 1968).

### Repertory Grid Theory

Kelly (1955, 1970, as cited in Latta & Swigger, 1992) developed Repertory Grid Theory from his Personal Construct Theory that he used in psychotherapy. Because it involves rating stimuli along a variety of featural scales, it can also be used to cluster stimuli and generate similarity matrices. The technique has been widely used in fields of expert system development and decision-

modeling (Batty & Kamel, 1995; Latta & Swigger, 1992; Nicholson, 1992), as well as in areas of marketing that are interested in discovering the features of a consumer experience (e.g. tourist packages or retail sales floors) that are important to the consumer (Mansfeld & Ginosar, 1994; Scott, 1993).

To construct a repertory grid of a collection of stimuli, various triads of stimuli are selected and shown to a subject. For each triad, the subject chooses one of the stimuli to be the "odd man out," i.e. the stimulus that doesn't quite fit with the others. The subject is then asked to supply a description of the difference, and a description of the similarity of the remaining two stimuli. These descriptions are called the two *poles* of a feature of the stimuli called a *construct*. After an unspecified number of constructs have been created, the subject goes through each stimulus and rates it on a scale between the poles of each construct. The results of the ratings form the repertory grid. The grid in Figure 8 offers a hypothetical example in the domain of vegetables. While the repertory grid approach asks subjects to define features explicitly, traditional similarity-based techniques such as MDS or clustering could be described as implicitly defining features, though the experimenter eventually attempts to define features based on the model.

	olive	potato	tomato	jalapeno	cucumber	zucchini	eggplant	bell pepper	
expensive	7	1	3	7	3	4	5	3	inexpensive
goes on pizza	7	1	7	4	1	4	4	7	not on pizza
often peeled	1	6	2	1	4	4	6	1	unpeeled
no seeds	1	7	7	7	7	7	1	1	has seeds
mushy	7	4	7	6	1	1	5	1	crisp

**Figure 8:** A hypothetical repertory grid in the domain of vegetables. The rows are the "constructs", and the columns are stimuli. Each stimulus is rated according to where it falls within the construct. In this example, a rating of 7 means the left characteristic is very appropriate, and 1 means the right characteristic is very appropriate.

In the table above, vegetables were rating higher if the description on the left was more true for them. The column of ratings for each vegetable could be then used as a vector in a five-dimensional coordinate space to establish distances (and presumably dissimilarities) between the vegetables. These proximities could then be used with MDS to determine whether the stimuli could be better described by a lower number of dimensions.

Batty & Kamel (1995) discuss various problems associated with repertory grid theory. Here we review some of these problems in comparison with TM. As in the case of the concept mapping technique, the main problem with repertory grid theory is its subjectivity. Although subjects are providing ratings, just like in traditional similarity-gathering experiments, the rating scales (constructs) are defined by each subject. Not only does this individual variance make it difficult to compare data across subjects, but it also allows for the creation of constructs that greatly vary in their level of abstraction, independence, and consistency of scale.

Some constructs will be binary or discrete, such as "has seeds/not", while others will be continuous, such as "mushy/crisp"; if the researcher intends to do typical MDS or clustering with the data, this mix of types can be problematic. Repertory grid theory has difficulty characterizing categorical features, even if they are not mixed with continuous features. A construct trying to describe the speed of a mammal's gait might likely have just three or four values in the rating scale: one for walk, trot, gallop, and perhaps canter. Because TM is by nature a discrete method, both types of features could be accommodated. Another problem with allowing untrained subjects to characterize the constructs is the difference between unipolar constructs, like "young/not young" vs. the bipolar construct "young/old". Depending on the domain, unipolar constructs can easily appear. Batty & Kamel maintain that such constructs violate the assumptions of the method, and suggest that they should be disallowed. Lastly, a repertory grid often has problems with features that span more than one dimension, such as color or musical timbre. Poles generated from such higher-dimensional features would be frustrating to use as the basis for a rating scale. The poles "red" and "green" could likely appear from the method of creating constructs through triads, but it would not be very informative to represent the colors of other objects as a rating between just these two colors.

We suggest that for the purposes of modeling a mental representation of organized stimuli, TM offers many of the advantages of repertory grid theory without the limitations stemming from its subjectivity and assumptions. The variety of feature types that TM allows is larger, and the fact that the subjects do not bias their answers by defining their own features likely leads to more robust solutions.

## Part IV: Chapter Summary & Thesis Overview

Trajectory Mapping (TM) has been compared with other similar scaling techniques: Pathfinder, NETSCAL, MAPNET, Concept Mapping, and Repertory Grid Theory. Although all methods except the last use a connected graph as their representational form, TM distinguishes itself through the form of its data; instead of modeling similarity values or high-level associations, TM subjects give orderings of stimuli. Combined with Part II, this comparative theoretical analysis of TM defends its role in the field as a method which complements hierarchical clustering and MDS in a different way than the methods described above.

In Chapter 2, we offer a data-driven comparison of MDS, hierarchical clustering, and TM. We present the results of using each model on experimental data from a variety of domains to illustrate the advantages and disadvantages of each approach.

Chapter 3 focuses on an algorithm for processing TM subject data and its assumptions. This algorithm has been created in part as a model of the manual technique of trajectory map construction suggested by Richards & Koenderink (1995). We also provide several diagnostic measures that can be used to compare the success of a model and the different trajectory maps that may result from different subjects.

Chapter 4 runs the algorithm through its paces and compares its resulting trajectory maps with trajectory maps generated manually. In the process, we introduce a variety of new data sets.

Chapter 5 analyzes representations in general, taking advantage of the TM technique to suggest that the traditional description of representations as either diagrammatic or sentential (Larkin & Simon, 1987) may need to be amended.

In Chapter 6, we summarize. The appendices that follow discuss theoretical work still in progress, several TM data sets that require further analysis, and some of the computer code used in the TM algorithm.

### References

- Augustson, J.G. & Minker, J. (1970) An analysis of some graph theoretical cluster techniques. *Journal of the Association for Computing Machinery*, 17(4), 571-588.
- Batty, D. & Kamel, M. S. (1995) Automating knowledge acquisition: A propositional approach to representing expertise as an alternative to repertory grid technique. *IEEE Transactions on Knowledge and Data Engineering*, 7(1), 53-67.



- Blais, C. (1993) Concept mapping of movement-related knowledge. *Perceptual and Motor skills*, 76, 767-774.
- Collins, A.M. & Loftus, E.F. (1975) A spreading activation theory of semantic processing. *Psychological Review*, 82, 407-428.
- Cooke N.M., Durso F.T. & Schvaneveldt (1986) Recall and measures of memory organization. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 12(4), 538-549.
- Cooke, N.J. (1992) Predicting judgment time from measures of psychological proximity. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 18(3), 640-653.
- Corter, J.E. (1996) *Tree Models of Similarity and Association*. Sage university paper series: Quantitative applications in the social sciences, no. 07-112. Thousand Oaks, CA: Sage Publications.
- Duda, R.O. & Hart, P.E. (1973) *Pattern classification and scene analysis*. New York: John Wiley & Sons.
- Gentner, D. & Stevens, A. (Eds.) (1983) *Mental models*. Hillsdale, NJ: Erlbaum.
- Harary, F. (1964) A graph theoretic approach to similarity relations. *Psychometrika*, 29(2), 143-151.
- Henderson, L., Patching, W. & Putt, I. (1994) Interactive multimedia, concept mapping, and cultural context. *Proceedings of AACE ED-MEDIA 94*, Vancouver, 269-274.
- Hutchinson, J. W. (1989) NETSCAL: A network scaling algorithm for nonsymmetric proximity data. *Psychometrika*, 54(1), 25-51.
- Johnson-Laird, P. (1983) *Mental models*. Cambridge, MA: Harvard University Press.
- Klauer, K. C. (1989) Ordinal network representation: Representing proximities by graphs. *Psychometrika*, 54(4), 737-750.
- Klauer, K.C. & Carroll, J.D. (1989) A mathematical programming approach to fitting general graphs. *Journal of Classification*, 6, 247-270.
- Klauer, K.C. & Carroll, J.D. (1991) A comparison of two approaches to fitting directed graphs to nonsymmetric proximity measures. *Journal of Classification*, 8, 251-268
- Kruskal, J. B. (1964) Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29, 28-42.

- Kruskal, J.B. (1976) KYST2A algorithm. Bell Laboratories, Netlib repository of software. <http://netlib.bell-labs.com/netlib/mds>.
- Larkin J.H. & Simon, H.A. (1987) Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, 65-99.
- Latta, G.F. & Swigger, K. (1992) Validation of the repertory grid for use in modeling knowledge. *Journal of the American Society for Information Science*, 43(2),115-129.
- Mansfield, Y. & Ginosar, O. (1994) Evaluation of the repertory grid method in studies of local's attitude towards tourism development processes. *Environment and Planning A*, 26, 957-972.
- Marr D. (1982) *Vision*. San Francisco: W.H. Freeman & Co.
- Minsky, M. (1975) A framework for representing knowledge. In P. Winston (Ed.), *A psychology of computer vision* (211-277). New York: McGraw-Hill.
- Newell, A. (1973) Production systems: Models of control structure. In W. Chase (Ed.), *Visual information processing*. New York: Academic Press.
- Nicholson, C. (1992) Learning without case records: a mapping of the repertory grid technique onto knowledge acquisition from examples. *Expert Systems*, 9(2), 79-87.
- Novak, J.D. & Gowin, D. B. (1984) *Learning how to learn*. Cambridge: Cambridge University Press.
- Pruzansky, S., Tversky, A., & Carroll, J.D. (1982) Spatial vs. tree representations of proximity data. *Psychometrika*, 47(1), 3-24.
- Quillian, M.R. (1968) Semantic memory. In M. Minsky (Ed.), *Semantic information processing* (216-270). Cambridge, MA: MIT Press.
- Richards, W. & Koenderink, J. J. (1995) Trajectory mapping: A new nonmetric scaling technique. *Perception*, 24 , 1315-1331.
- Rumelhart, D.E. & Ortony, A. (1977) The representation of knowledge in memory. In R.C. Anderson, R.J. Spiro, & W.E. Montague (Eds.), *Schooling and the acquisition of knowledge* (99-135). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Schank, R.C. & Abelson, R.P. (1977) *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Hillsdale, NJ: Lawrence Erlbaum Associates.

- Scott, C. (1993) Semiotics of retail space: An application of the repertory grid methodology. *Semiotica*, 94(3/4), 295-303.
- Shepard, R. N. (1962) The analysis of proximities: Multidimensional scaling with an unknown distance function. I. & II. *Psychometrika*, 27, 125-140, 219-246.
- Shepard, R.N. (1974) Representation of structure in similarity data: Problems and prospects. *Psychometrika*, 39, 373-421.
- Sherratt, C.S. & Schlabach, M.L. (1990) The application of concept mapping in reference and information services. *RQ*, 30(1), 61-69.
- Smith, E.E. (1989) Concepts and induction. In M.I. Posner (Ed.), *Foundations of cognitive science*. Cambridge, MA: MIT Press.
- Tolman, E.C. (1948) Cognitive maps in rats and men. *Psychological Review*, 55, 189-208.
- Torgerson, W. S. (1952) Multidimensional scaling: I. Theory and method. *Psychometrika*, 4, 401-419.
- Tufte, E.R. (1990) *Envisioning Information*. Cheshire, Connecticut: Graphics Press.
- Tversky, A. (1977) Features of similarity. *Psychological Review*, 84(4), 327-352.



## Chapter 2

### A Data-Driven Comparison of TM with MDS and Hierarchical Clustering

#### Abstract

To illustrate how TM complements the current array of scaling techniques, we illustrate the use of multi-dimensional scaling, hierarchical clustering, and TM on three different data sets. We demonstrate that TM can offer valuable inferences about data that other methods would neglect, and that TM is particularly useful for more abstract, conceptual stimuli.

#### Introduction

In this chapter we model subjects' conceptual representations of various domains using each of the three modeling techniques, MDS, clustering, and TM. Our goals are to illustrate the advantages of each approach and to derive a broad rubric for advising the use of the three methods. There are two basic means of comparing these methods: we could devise quantitative measures that choose one technique over another based on an independent cross-approach measure of data explained, or we could quantitatively examine the degree to which each technique explains a data set that we already understand. Pruzansky, Tversky, & Carroll (1982) take the first approach in their paper comparing MDS and clustering. Because it is difficult to develop an independent fitness measure for all three techniques, however, we take the latter approach. We examine two familiar domains, Boston subway stations and musical intervals, and one less familiar domain, a collection of sound textures. Success in the case of novel domains can be qualitatively measured by the number of "sensible" features that the model suggests.

We chose these data sets because they represent a range of cognitive abstraction. Subways stations are a conceptual stimulus; each represents a physical location as well as an experience. Stations can have many features associated with them. Musical intervals are definitely perceptual stimuli; each interval can be identified as one of 12 intervals which Western listeners are used to hearing whether they were conscious of the names of the intervals or not. Finally, sound textures are close to purely sensorial stimuli, since although one can name the causal process behind the sound, the sound textures themselves are not something for which we have explicit perceptual categories. We will not be exploring each data set fully; we want mainly to offer some concrete examples that illustrate the differences in methods. For each domain we explore the data of one subject.

#### Subway Stations

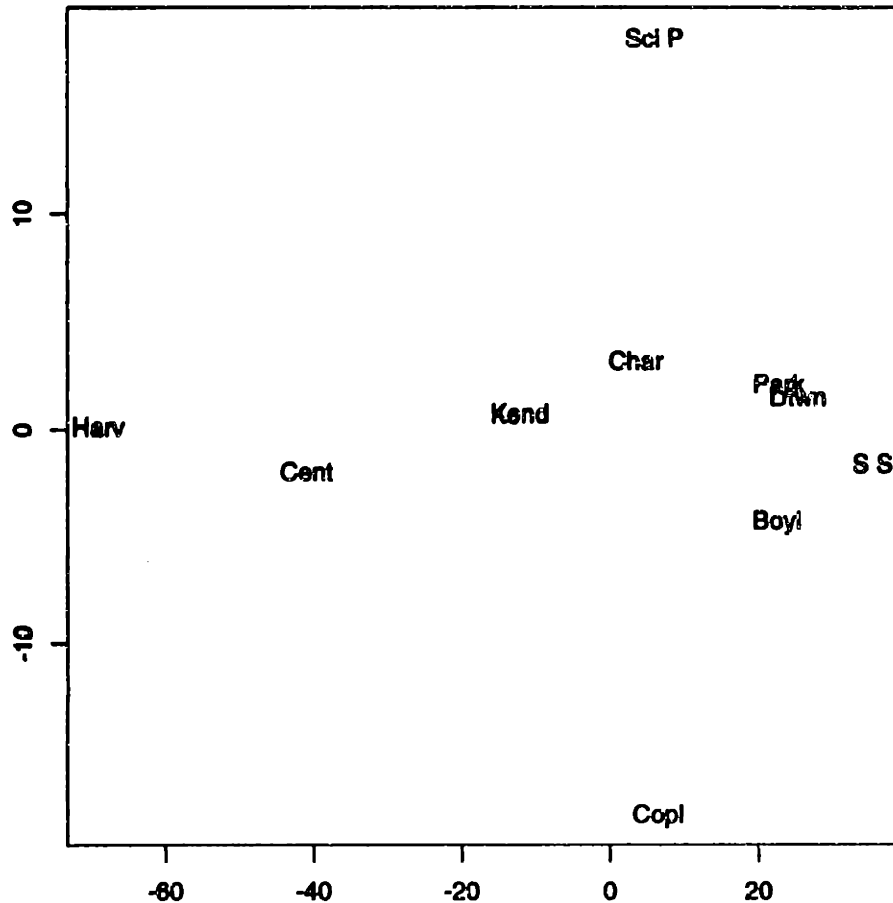
A set of 10 subway stations of Boston is a relatively abstract domain. Although there are certainly domains that could be thought of as more abstract or conceptual, such as historic events or tourist attractions, subway

stations qualify as abstract because they can be thought of in several contexts. They have several levels of features that must be understood for the subject to grasp all of these contexts. Where are the stops in relation to each other? Where are the stops in relation to the city above them? Are the stops on the same train line? Note how each of these questions is answered best by a different representation.

The subway stations in this example are Boylston, Central, Charles St., Copley, Downtown Crossing, Harvard, Kendall, Park St., Science Place, and South Station.

Figure 1 is an MDS plot that comes from asking a subject for the “geographic similarity” of each pair of stations, e.g. “How near is this station to that station, on a scale from 1 to 10?” An algorithm called Multi-Dimensional Scaling (MDS) can turn these data into a metric space. The space illustrates quite well the geographic aspect of subway station knowledge, where the stations are in relation to each other. The dimensions of the space could be vaguely described as North-South and East-West, although they appear slightly skewed here.

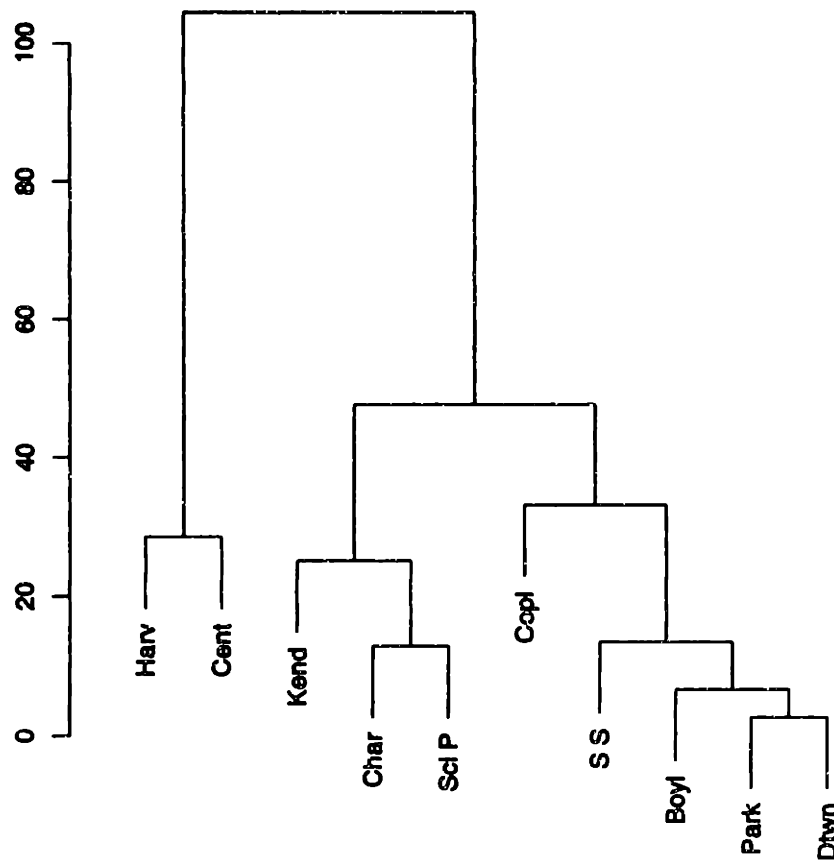
Note, however, that this plot tells us nothing about which stops are connected to each other (the “routes”), or where city boundaries lie (the “clusters”).



**Figure 1:** This MDS plot of the similarity data for the subway stations illustrates quite well the geographic nature of subway station knowledge. The dimensions (though somewhat warped as described by Lynch, 1960) are similar to the North-South-East-West axes of a traditional map of the Boston area.

Figure 2 is the tree of hierarchical clusters representing the same similarity data as Figure 1. Here, the stations can be seen to be clustered by geographic region. The pair of stations in lower right corner represent the core of downtown Boston. The four right-most stations make up a broader downtown area; the five right-most stations are those in main Boston area. The Kend-Char-Sci P cluster are those along the Charles river, and the left-most two stops are those deep in Cambridge. These are the groupings that this particular subject makes when he thinks about the subway stops. These clusters answer the question, "Where are the stops in relation to the city?"

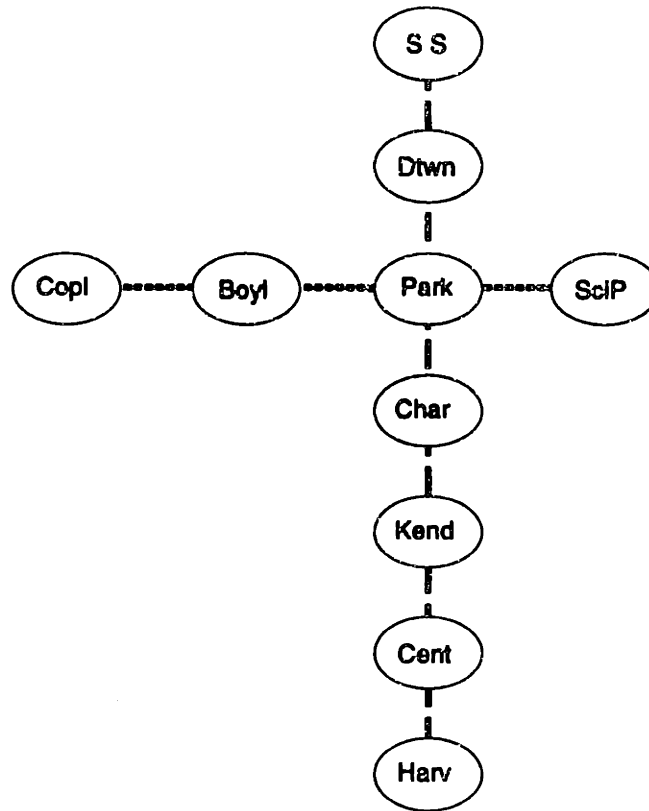
Note that we still don't know about which stations are on the same lines.



**Figure 2:** This hierarchical cluster tree of the subway station data reveals clusters based on geographic region. The pair of stations in lower right corner represent the core of downtown Boston. The four right-most stations make up a broader downtown area; the five right-most stations are those in main Boston area. The Kend-Char-Sci P cluster are those along the Charles river, and the left-most two stops are those deep in mid-Cambridge.

Figure 3 shows the trajectory map of the subway data. By overlaying the orderings of subject data over the graph, we can see which nodes form trajectories. In this case, there are two trajectories that overlap at the Park station. This graph reveals the subway lines themselves, something neither of the other approaches could do. The nodes running from S.S. to Harv are on the Red Line, and Copl through SciP are on the Green Line. The Park node, as it is indeed shown, is an intersection point where riders may change trains. Our clean division of clusters likely stems from the subject mentally moving from station to station along single train lines. If the subject lived at Copley, however, and traveled to Charles St. for work, then the data would have led us to see Park as a node where the paths “turn corners”, and we may not have found the division of train lines. Nevertheless, domains with stimuli that can be thought of as varying along separate but overlapping clusters, like subway stations, are very appropriate for TM.





**Figure 3:** This trajectory map of the subway data reveals the subway lines themselves, something neither of the other approaches could do. The nodes running from SS to Harv are on the Red Line, and Copl through SciP are on the Green Line.

For this domain, each scaling technique contributes a different insight into the data. MDS illustrates where the stops are in geographic relation to each other. The hierarchical clustering tree groups stops together that have relevance to their area in the city. TM shows which stops are connected to each other, and in which order. This comparison illustrates that each method can play a part in the analysis and that TM complements the traditional methods well.

### **Musical Intervals**

As an example of the perceptual level of stimuli, we choose the domain of musical intervals. Data here were collected for Gilbert and Richards (1994). We consider the domain of musical intervals to be slightly more abstract than a purely psychophysical domain such as sound textures or colored lights; unlike these lower-level domains, which are simply percepts, musical intervals are stimuli which involve a previously learned structuring system (that of the music in our society) that will influence the way subjects consider the stimuli no matter what their musical ability. The question thus becomes, how do subjects organize their knowledge of musical intervals? Can the scaling techniques reveal what the subjects might have learned from their environment of Western music?

Musical intervals are made up of simply two simple tones. For musicians, the relationship of the tones stimulates a learned percept such as "major 3rd." In the figures below, stimuli are designated with a small "m" to indicate "minor" and a large "M" to indicate "major". "P" indicates "perfect". Thus, the intervals are, in order of increasing size, minor 2nd, major 2nd, minor 3rd, major 3rd, perfect 4th, tritone, perfect 5th, minor 6th, major 6th, minor 7th, major 7th, and octave. Subjects used a Hypercard software interface with buttons that played the various intervals. At each trial, the subject would see three buttons, labelled "A," "B," and "X" and be asked, "Is A or B more harmonically similar to X?" These data were transformed to a similarity matrix according to the Method of Triads (Torgerson, 1952).

Figure 4 shows the MDS plot for similarity judgments of a non-musician subject. The Octave and Perfect Fifth share almost the same coordinates. We find again that the MDS plot of similarity judgments gives us some salient features along each of the dimensions. The Y axis increases with the size of the interval. The X axis increases with the Western tonality of the interval. (The major 3rd, octave, perfect fifth, and perfect fourth are traditionally very common in tonal Western music.)

The tree plot in Figure 5 of the hierarchical clustering of the musical intervals similarity data reveals the same features as MDS. The top clusters divides the intervals by size, and the subclusters group stimuli of similar tonality. The tree is more helpful than MDS at visualizing which smaller subsets of stimuli belong together. The perfect fifth and the octave, arguably the two most salient harmonic structures in Western music, are matched. The major 3rd and the perfect 4th, two other harmonically important intervals, are also paired. We also see that the minor 2nd and the tritone, two notoriously dissonant intervals, are clustered together.

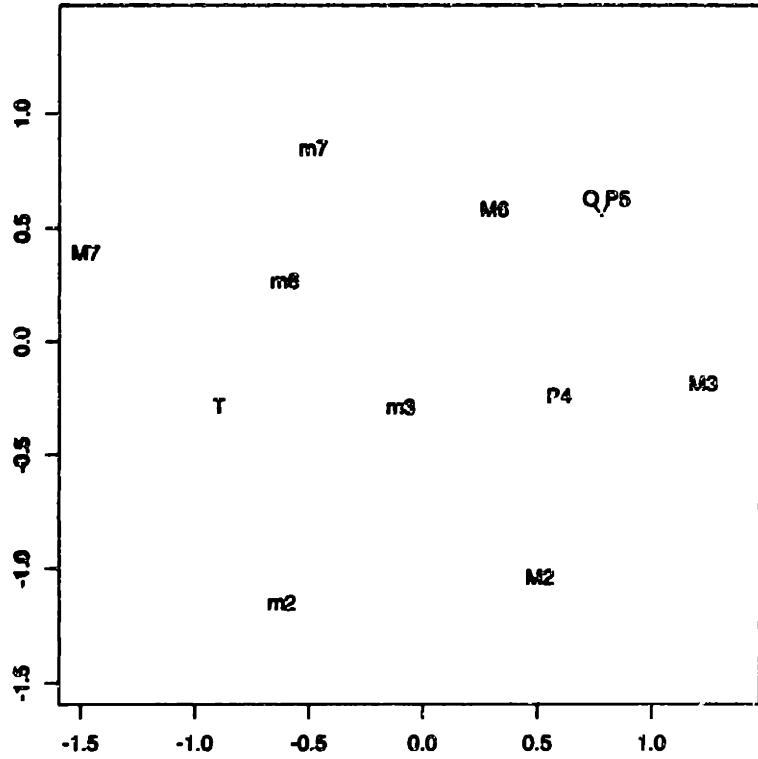


Figure 4: MDS plot of similarity judgments for musical intervals. The y-axis increases with the size of the interval, and the x-axis increases with the Western tonality of the interval.

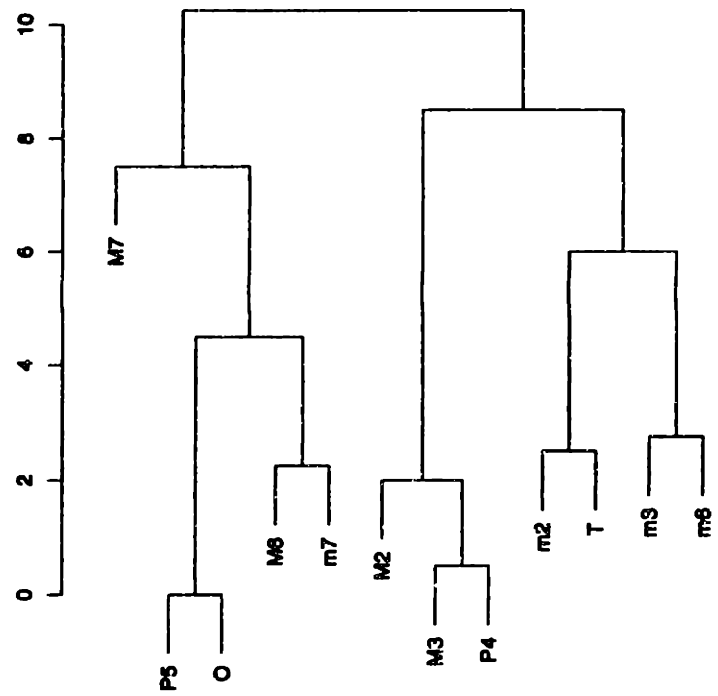


Figure 5: A tree plot based on hierarchical clustering of the musical intervals similarity data. The top division groups the intervals by size, and the subclusters group them by similar tonality.

Figure 6 illustrates the trajectory map of musical intervals for the same subject. For the TM task, subjects used a different Hypercard interface with buttons for each interval. In each trial, two intervals were chosen out of the 12, and subjects were told to “think of a feature that varies across the two intervals. Pick an extrapolant by choosing an interval that would best continue that change.” By overlaying the orderings of subject data over the subject’s graph, we find three distinct trajectories that overlap. The first trajectory runs from m2 to M3. The second runs from M3 to m6, and then branches to m7 and M6. The small ⊗ sign indicates that no data run along the nearby branches, e.g. (m7, m6, M6). The last trajectory runs from m7 to O. The nodes of the graph are arranged to highlight the breaks between the trajectories. Links are also hatched accordingly.

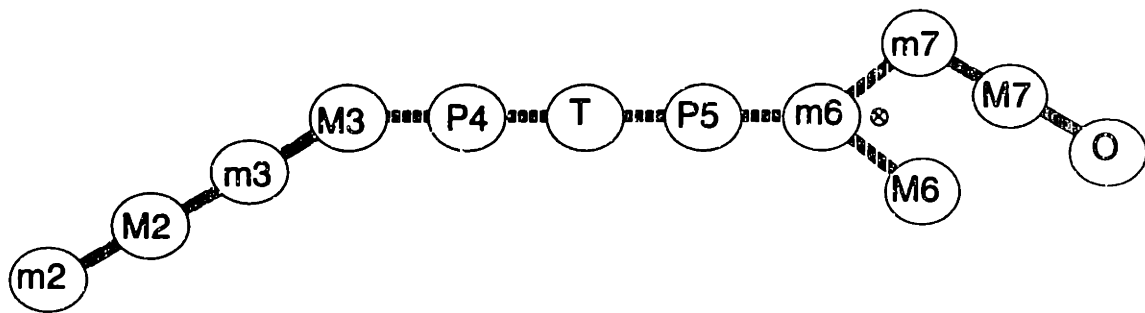


Figure 6: Trajectory map of the musical intervals. Trajectory clusters distinguish low, middle, and higher intervals, breaking at points probably based on the tonal relations of each group.

The trajectories can be seen to represent clusters of smaller intervals, medium intervals, and larger intervals. The ordering of the trajectories themselves, as well as the stimuli, show that the subject was using size, the most salient feature of intervals, to order them. The boundaries between trajectories, however, illustrate some intuitions about tonality. The intervals at each end of the scale, such as the 2nds and the 7ths, sound much more dissonant when played one after another than those in the middle. In the middle we find the building blocks of more tonal harmony, the major 3rd, the perfect 4th, the perfect 5th, the major 6th, and the minor 7th. (The minor 7th is often present in chords, though when played alone, especially in contrast other nearby intervals, it sounds strongly dissonant.)

In this domain, in which there are basically two features, interval size and degree of harmony, each technique sheds similar light on the data. The harmonic feature is complex, however, as we see in the confusing tree. Because TM is based on sequences, the trajectory map represents mainly the feature of interval size and offers only small indications of harmonic preferences. In this domain, MDS seems to be the best representation, given that our criteria at this level of comparison is based on the degree to which the representation would illustrate the features we know are important to an experimenter who did not have as much prior knowledge.

## Sound Textures

As an example from the sensory domains, we illustrate below each of the scaling methods with sound textures. Sound textures are sounds that play a similar role as visual textures; they form a recognizable patterned percept not through an individual aspect of the texture, but through its periodic repetition. Sound textures are often what the lay person would call "background noise". In an experiment done with Saint-Arnauld (1995), we collected similarity data and TM data on 10 sound textures. Each of the plots below illustrates the data of the same individual.

In both experiments subjects used digitized sounds represented by icons in a Macintosh interface. To calculate similarities, subjects were given a series of screens with one sound icon (A) set apart from the others. Subjects were told to choose the sound most similar to A from the set remaining and drag it toward A. Subjects then chose the next-most similar sound to A, etc., until all sounds were ordered in terms of similarity to A. This procedure was repeated for each sound. To collect TM data, subjects were presented with a pair of sound icons and the set of icons for the remaining sounds. Subjects were told to "listen to the two reference sounds, and choose a characteristic which differentiates the two. From the rest of the set, try to find a sound that fits between/would go after/before the reference sounds, according to the characteristic you choose." Subjects dragged the appropriate icons in place as the extrapolants and interpolant.

The 10 sound textures are *whispers* (many people whispering), *crickets* (hundreds of crickets chirping), *wind*, *A/C* (the droning hum of an air conditioning unit), *stream* (the flowing water), *snare* (a snare drum roll), *bubbles* (water bubbling), *crowd* (the noise of a crowd milling and talking), *applause* (the sporadic clapping of a small crowd), and *traffic* (car engines and honking horns). Each sound is approximately five seconds long.

Figure 7 is an MDS plot based on the similarity judgments. It reveals two feature axes quite nicely; the X axis can be seen to be irregularity of the period of the sound textures, e.g. the period of sporadic applause is more irregular than the motor of an air conditioning unit. The Y axis can be seen to be the distinctiveness of the beats within the texture; as the axes increases, the texture beats become more subtle. The snare drum and air conditioning unit have distinct beats, but the whispering texture (many people whispering) flows quite smoothly; it would be difficult to say where the period begins or ends.

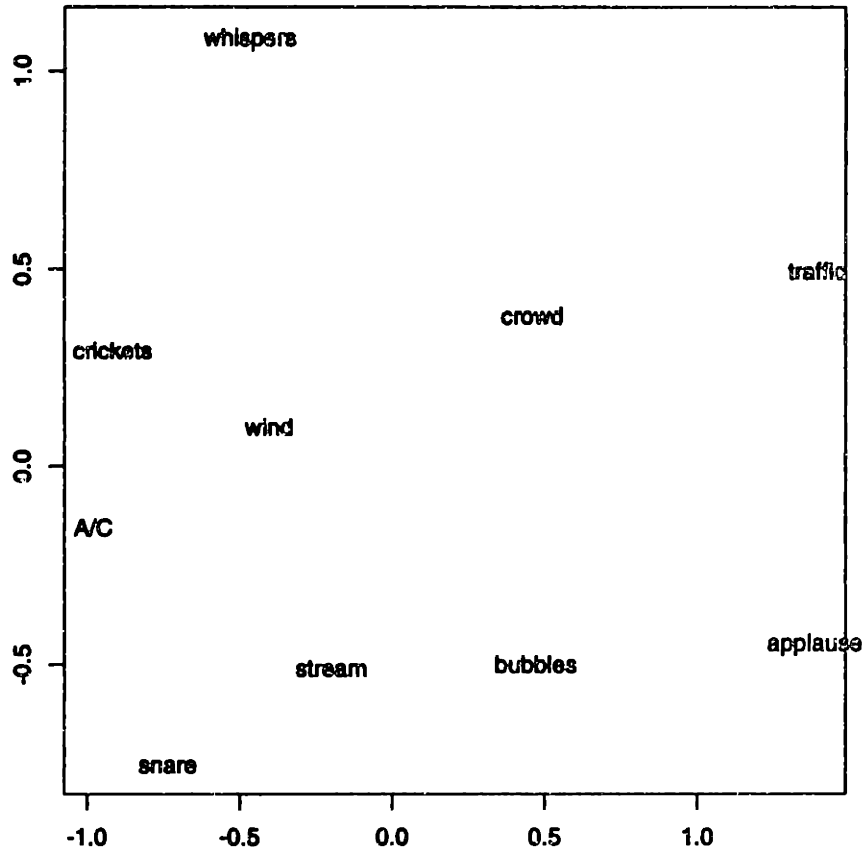
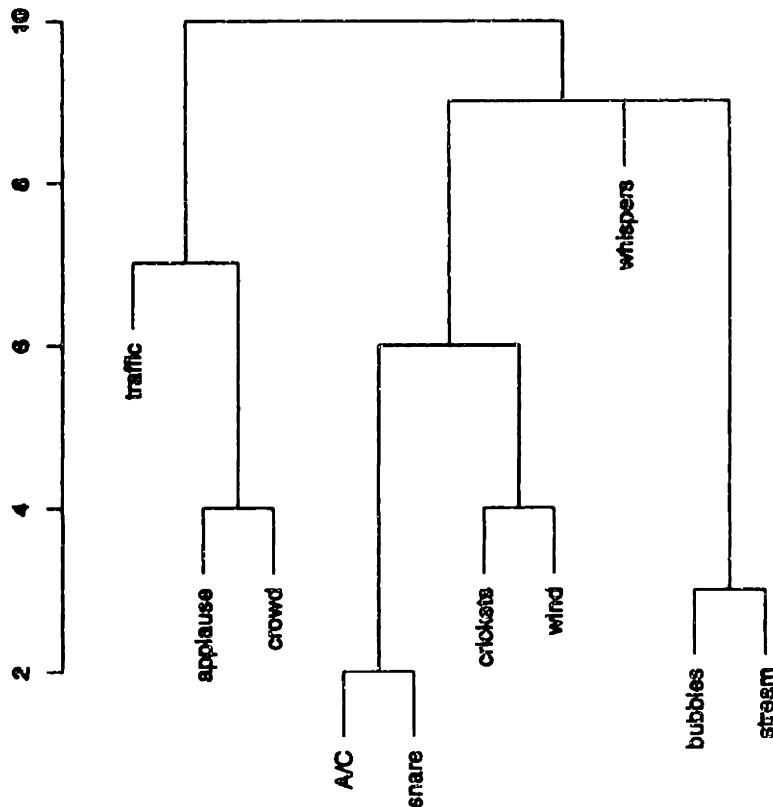


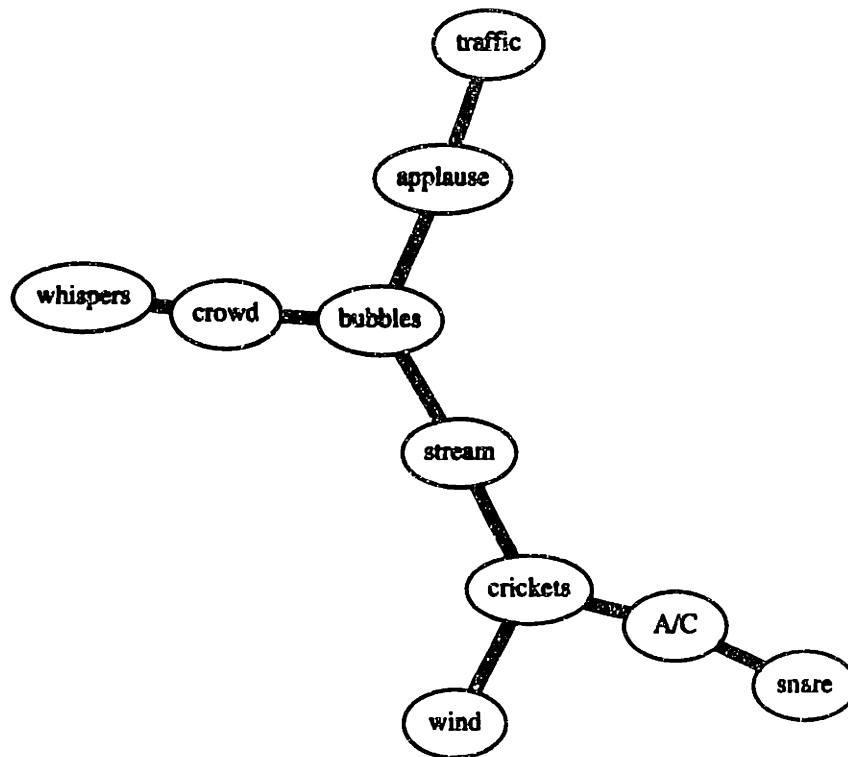
Figure 7: MDS plot based on one subject's similarity judgments on sound textures. X axis can be seen to be irregularity of the period of the sound textures; the Y axis can be seen to be the distinctiveness of the beats within the texture.

Figure 8 shows a hierarchical clustering tree of the sound texture data. The stimuli can be seen to be organized again roughly by aspects of periodicity. The two largest clusters group textures according to whether they are periodic. The *traffic*, *crowd*, and *applause* sounds have arbitrary outbursts of horns, shouts and bursts of claps. The other cluster contains periodic sound textures, seemingly sub-clustered by the degree of their regularity. The air conditioning unit and the snare drum are matched, their both being generated by an regular process. The two water sounds are also paired, having a similar generating process and therefore similar levels of regularity. It is unclear why *crickets* and *wind* are clustered together.



**Figure 8:** This hierarchical tree clustering of the sound texture similarity data. Two major clusters divide periodic and aperiodic textures. Periodic textures may be grouped by regularity of period, e.g. *A/C* and *snare* are both very regular.

Figure 9 contains the trajectory map of the sound textures. The map illustrates some of the same information as both the MDS and cluster plots. By looking at locally neighboring nodes, we can see how the nodes cluster, and by looking at the pathways, we can see some of the features as in the MDS plot: periodicity and regularity of period. The aperiodic sounds lie at the top end. Near the bottom lie the most regular periodic sounds, with the regularity of the period decreasing as one moves up the graph (away from snare and wind). Moving from *whispers* through *bubbles* to *traffic* we see a path perhaps representing the increasing distinctiveness of the beats in all of these aperiodic sounds. Because the triplet data reveal no separate clusters in the trajectory map, we can infer that the features moving along the trajectories are reasonably complex, i.e. it is possible to consider the stimuli lying in the middle of the trajectories as belonging to several contexts.



**Figure 9:** The trajectory map of the sound textures shown above illustrates knowledge from both of the MDS and Tree plots. Aperiodic sounds are near the top, very regular periodic at the bottom. The regularity of the period decreases along the graph (away from *snare* and *wind*). There are no separate clusters in this trajectory map.

Note that MDS, hierarchical clustering, and TM all offer information about the periodicity of the stimuli, and the regularity of their patterns. The tree is not a particularly good representation, however, because it doesn't give an idea of the continuity and variability of the features. MDS and TM do offer this variability. TM gives a reasonable impression of the multidimensionality of the data through the branches which are not separate clusters. MDS also gives an idea of the dimensionality, but lacks the local connections. Because each representation offers a valuable aspect of the information, the most informative representation in this case would be a combined plot of the trajectory map over the MDS data.

### Summary of Part I

We have presented three representations of data from the domains of subway stations, musical intervals, and sound textures. For the subway station data, it appeared that each representation offered a different type of information about the data, suggesting that all three methods should be used to maximize the return of information. For musical intervals, MDS offered the solution most consistent with our prior knowledge: the data have only two salient features, and because the features were continuous, they could be well represented along the dimensions of the metric space. For sound textures, because all the features of the data seemed to make little sense in discrete clusters, MDS and TM were the better techniques. We compare the three



methods at a more theoretical level below, and in Chapter 4 we describe further trajectory maps from variety of other data sets.

## PART II: TM Orderings vs. Similarity

We will now briefly discuss the relationship of TM data (orderings) and similarity data. Since older methods are largely based on similarity, it would be noteworthy if similarities could be derived from either the TM subject data or from the Trajectory Maps themselves.

### Adjacencies

A matrix of proximities can indeed be derived from TM data. A value  $x_{ij}$  of the matrix is the number of times that stimulus  $i$  appeared next to stimulus  $j$  in the quintuplet data. We call these values the *adjacencies* of the subject data. One might then wonder how similar the TM adjacencies matrix for a data set would be to a similarity matrix gathered from the same subject.

To explore this issue, we offer data from seven subjects from four different data sets. At separate sittings, each subject gave both similarity values for each stimulus pair and TM quintuples. For this comparison, we assume symmetrical matrices for both the similarity matrices and the adjacencies matrices. We offer below a table of Spearman's rank correlation on vectors made up of the rows of the upper triangles of each matrix. All correlation values were significant with  $p < 0.01$ ; for most,  $p$  was several orders of magnitude less. Note that there is almost always a correlation of at least 0.4, between the matrices of the same subject, often as high as 0.7.

Data Set	Rank Correlation of Similarity and Adjacency Matrices
Boston Tourist Sites A	0.66
Boston Tourist Sites B	0.43
Circles A	0.82
Circles B	0.54
Musical Intervals A	0.33
Musical Intervals B	0.58
Sound Textures A	0.46
Subway A	0.63

Why are the matrices not more correlated, and why are the correlations lower on some data sets than others? We suggest that correlations will be lower for data sets in which stimuli can be easily sequenced according to features that would not typically be used in judging similarity. For example, in the case of musical intervals, intervals can be easily sequenced by their size, but subjects likely judge their similarity based on any of several harmonic relations. Likewise in the case of the Boston tourist site data, subjects may sequence sites based on the order they might visit them, or their physical proximity to each other while reporting similarity values based on thematic similarity.

### **Manipulations Allowed on Each Representation**

Our choice of mental representation can make an enormous difference in the way that we analyze a problem, conduct a search, or convey instructions or ideas to a colleague (Larkin & Simon, 1987; Winston, 1980; Norman, 1993). A good train schedule, for example, allows a traveler to discover what the best connecting trains would be for her particular trip. We offer here a brief analysis of metric spaces, hierarchical clusters, and trajectory maps in these terms, describing some of the advantages of each representation.

Metric spaces make it easy to see which stimuli are most similar, and least similar, to a chosen stimulus. One can simply look at the distance from the chosen stimulus. Assuming that one has meaningful labels on the dimensions, the metric space also gives an indication of *how* two points differ. If one would like to learn where a novel stimulus would fit in the space, knowing only the similarity between the novel stimulus and any one current stimulus constrains the location significantly (to a circle). Knowing the relationship of two current points to the novel stimulus constrains the novel's location further (to two points), and knowing three similarity values anchors it. Learning more global information, however, such as how many stimulus grouping there are, or whether there are significant relationships that span more than two stimuli, requires clustering or TM.

A tree representing hierarchical clusters enables simple checking on which stimuli belong to the same grouping as a chosen stimulus. One can gain a vague idea of how similar any two stimuli are by looking at how far away they are in the tree, but this graphic information can be deceptive because of the non-determined order of tree branches. The hierarchical nature of the clusters makes it easy to see which stimuli share many features with each other, which share fewer features with each other, and which share few or no features. If one would like to place a novel stimulus in the tree and doesn't mind simply appending a branch onto to an existing cluster (as opposed to rearranging to tree to remain binary, etc.), then one can sometimes include the novel stimulus with just one similarity relation to a current stimulus (i.e. when it is very similar to a current stimulus). Other times, one might require numerous similarity relations to place the novel stimulus appropriately (i.e. when the relations one has are all large dissimilarities from stimuli in one branch of the tree). Note that it is difficult from a tree to know *how* two stimuli are different, even when you know their similarity value.

A trajectory map allows the researcher to see the order in which stimuli can be sequenced. Because the sequencing was inevitably done by the subject according to some feature, and because similarity judgments must be done by examining features, one can read a measure of stimulus similarity from the orderings, though the notion of similarity is slightly weaker than in metric spaces or hierarchical clustering. Like in clustering, if one would like to place novel stimulus into the trajectory map, the amount of information required can vary. If one knows that the novel stimulus can be sequenced directly between two neighboring stimuli in the trajectory map, then it can be placed. Likewise, if one knows that the novel stimulus can be sequenced between two stimuli on trajectories that otherwise do not intersect, then it is necessary to create a new trajectory that spans them and incorporates the novel stimulus. If one knows only information from more distant stimuli on the same trajectory, however, more information is required. Like a metric space, however, one can usually tell *how* any two stimuli in the trajectory map are different.

### Part III: Conclusions

Having seen the different types of information offered by metric spaces, hierarchical clusters, and connected graphs, it seems not unreasonable to explore each representation when exploring an unfamiliar domain. Indeed, had we not explored the subway stations using TM, we would not have known that there were important connections between them that could not always be inferred from their geographic layout. Similarly, without looking at the MDS plot for the musical intervals, we would not have known how similar the Octave and Perfect Fifth were (the tree and trajectory map had no relevant ordering).

Technique	Type of features best suited for technique
MDS	continuous features, independent
H. Clustering	categorical features, hierarchical
TM	continuous & categorical features that sequence stimuli

**Table 1:** The three techniques and the types of features that they best model.

While we have not offered quantitative measures to choose between MDS, clustering, and TM, our comparisons have illustrated some basic tendencies of each method. Table 1 lists the type of features that are best modeled by each method. If stimuli are expected to have features that might order them in semi-linear fashion, then TM should be performed. If the stimuli are known to have no salient discrete features, then hierarchical clustering might be neglected, since it will not likely offer new information over the other two methods. Since TM can handle some multiple feature-contexts (through its overlapping trajectories), TM can be recommended for more abstract, conceptual stimuli that are likely to have many features. There is not always a

sure-fire treatment, however; the musical intervals and their complex feature of tonality illustrated that even TM breaks down in the face of features too complex to order easily.

In comparison to similarity-based methods, we believe that TM offers an important contribution to feature scaling by constructing a representation of stimuli based on a measure other than similarities. By giving subjects an opportunity to sequence the stimuli within the contexts of different pairs, rather than compare them two-by-two alone, TM can model distinct features that might get grouped into the generic feature "similarity" in other methods. Also, TM combines some advantages of both metric spaces and clustering: its semi-linear, rank-ordered clusters can model both continuous and categorical features.

## References

- Gilbert, S.A. & Richards, W. (1994) Using trajectory mapping to analyze musical intervals. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta.
- Larkin J.H. & Simon, H.A. (1987) Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, 65-99.
- Lynch, K. (1960) *Image of the City*. Cambridge, MA: MIT Press.
- Norman, D. (1993) *Turn signals are the facial expressions of automobiles*. Reading, MA: Addison-Wesley Pub. Co.
- Pruzansky, S., Tversky, A., & Carroll, J.D. (1982) Spatial vs. tree representations of proximity data. *Psychometrika*, 47(1), 3-24.
- Saint-Arnauld, N. (1995) *Classification of sound textures*. Masters Thesis, MIT Media Lab. To receive a copy email docs@mit.edu or call MIT Document Services at 617 253-5650.
- Torgerson, W. S. (1952) Multidimensional scaling: I. Theory and method. *Psychometrika*, 4, 401-419.
- Winston, P.H. (1980) Learning and reasoning by analogy. *Communications of the ACM*, 23(12), 689-703.

## Chapter 3

### An Objective Approach to Trajectory Mapping through Simulating Annealing

#### Abstract

Trajectory Mapping (TM) was introduced in 1995 as a new experimental paradigm and scaling technique. Because only a manual heuristic for processing the data was included, we offer an algorithm based on simulated annealing that combines both a computational approach to processing TM data and a model of the human heuristic used by Richards and Koenderink. We describe the details of the algorithm itself, present several diagnostic measures, and analyze the assumptions made by choice of the representation and the cost function.

#### Introduction

Psychologists have often explored how people organize their knowledge about different objects. A variety of experimental paradigms, such as Multi-Dimensional Scaling (MDS) and hierarchical clustering, have been used to try to elicit the features of stimuli that subjects find important.

In 1995 Richards and Koenderink proposed a new scaling technique called Trajectory Mapping (TM). Broadly speaking, TM is an experimental paradigm that asks subjects for judgments about the features of stimuli and then models the subjects' conceptual structure with a connected graph. Richards and Koenderink describe the paradigm generally and give several examples, but do not offer a detailed algorithm for deriving the graphs from the subject data. We hope to offer a complete experimental TM procedure by describing and analyzing such an algorithm. The graphs resulting from this algorithm resemble those of Richards & Koenderink previously made by hand using various heuristics.

#### Part I: Overview of the Approach

In this section we present the basic steps of going from subject data to a trajectory map. First, the quintuplets described above are broken down into triples. Of a quintuplet A - B - C - D - E, for example, the triples would be A - B - C, B - C - D, and C - D - E. The number of times each triple occurs in the quintuplet data is assigned as the weight on that triple. The weights are normalized by the maximum possible weight ( $n$  for  $n$  stimuli) so that the

maximum weight is 1.0. Alternate methods of counting triples have been tried and are described in Appendix A.

The triples are used to build graphs through overlaps (Figure 1). If triple (1, 5, 3) occurs frequently (has a high weight), and triple (5, 3, 4) occurs frequently, then we connect them to form the chain 1 - 5 - 3 - 4. If we could continue to connect triples until we formed a complete graph, then the process would be simple. Often triples conflict, however. There are two main ambiguities that must be resolved in the triple linking process. The first conflict is called a "split-or-fit." If our next heavily weighted triple is (5, 3, 2), for example, we aren't sure what to do with 4 and 2. We could have each branch from 3 (splitting the chain), or we could fit them both into the same straight chain, giving less importance to the coherence of one of the triples, e.g. 1 - 5 - 3 - 4 - 2.

Triples to satisfy	Resulting graph so far
(1, 5, 3)	
(5, 3, 4)	
(5, 3, 2)	<p style="text-align: center;">or</p> <p style="text-align: center;">or</p>
(5, 4, 3)	<p>??? (conflicting triple)</p>

**Figure 1:** Building a trajectory map from the triples can be difficult because of decisions about branching and conflicting triples.

The other type of ambiguity is simply called a "conflicting triple," i.e. when two triples suggest different ordering for the same three stimuli. In our example, (5, 4, 3) would be a conflicting triple since it contradicts (5, 3, 4).

Conflicting triples can indicate noisy subject data, since a subject who does not behave consistently would generate more conflicting triples, but they can also arise from two stimuli close enough to each other in the feature space that the subject considers the two orderings interchangeable. Usually conflicting triples cannot both be satisfied in a graph, although it sometimes makes sense to include them both by fitting them in a small triangular cycle.

Solving the problem of finding the best graph given the triples is a combinatorial optimization problem. Richards and Koenderink solve this problem through a manual heuristic that involves systematic trial and error, resulting in what is subsequently taken to be the “simplest” graph, given the data, while allowing for some inconsistent triples. We offer a TM algorithm that make this procedure objective, using a simulated annealing paradigm. Both the manual and the algorithmic methods convert a subject’s triples to a graph (Figure 2). We use a diagnostic measure of fit (described below) to determine a “noise level” in the data that we use to decide which triples to include in modeling the graph. We typically do not include triples with frequency 1 or 2, for example. The resulting graph is then examined; by thresholding the weakest links and tracking where the triples lie on the graph, salient subgraphs can be discovered. When these subgraphs can be seen as chains, we call them trajectories.

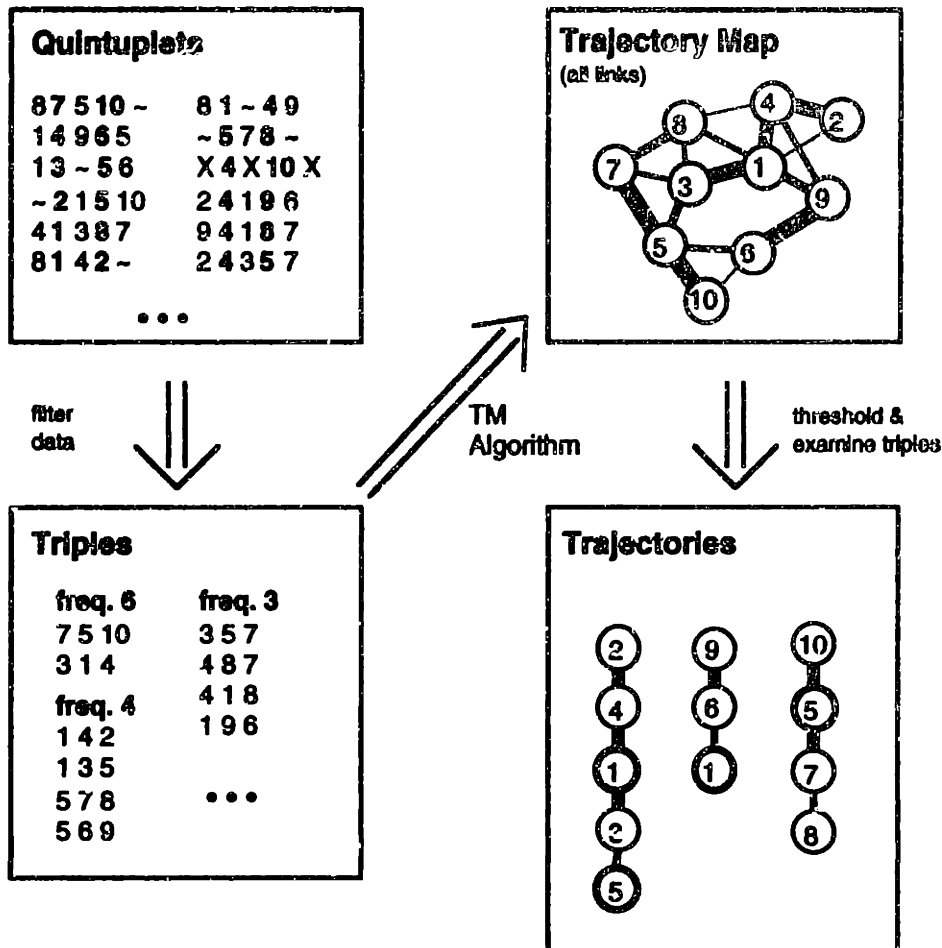


Figure 2: Summary of the Trajectory Mapping procedure. Note that nodes 1 and 5 appear in two trajectories, suggesting that two separate features (for each) are used to organize the data.

## PART II: The Algorithm

The goal of the algorithm, as described above, is to find the best possible graph as a model of the triples. In our approach, we begin by constructing an initial unit-link graph with all the links that would be necessary to satisfy all the triples. We then optimally adjust this graph for the given triples and for a certain cost function. This optimization takes place by carrying out simulated annealing using Gibbs sampling (Press, et al, 1988). The state variable that we optimize is a binary link-matrix, stochastically adding and removing links to minimize the cost function. After finding the optimal unit-link graph according to the annealing process, we then adjust the parameters of the cost function and begin again. After iterating this process over a large space of cost function parameters, we have a collection of unit-link graphs that are each optimal for their particular parameter settings. (We discuss below how the various parameters affect the optimal graphs.) To calculate the final trajectory map, with a range of weights on the links, we average the optimal unit-link



graphs over the space of the cost function parameters. See Figure 3 for an overview of the procedure.

It is important to note the two different spaces that we work in: the first is the space of all unit-link graphs, given a certain cost function (in which we find the optimal graph), and the second is the space of all reasonable cost functions (the optimal graphs of which we average across to find the final trajectory map). Next we present the cost function and a description of its parameters.

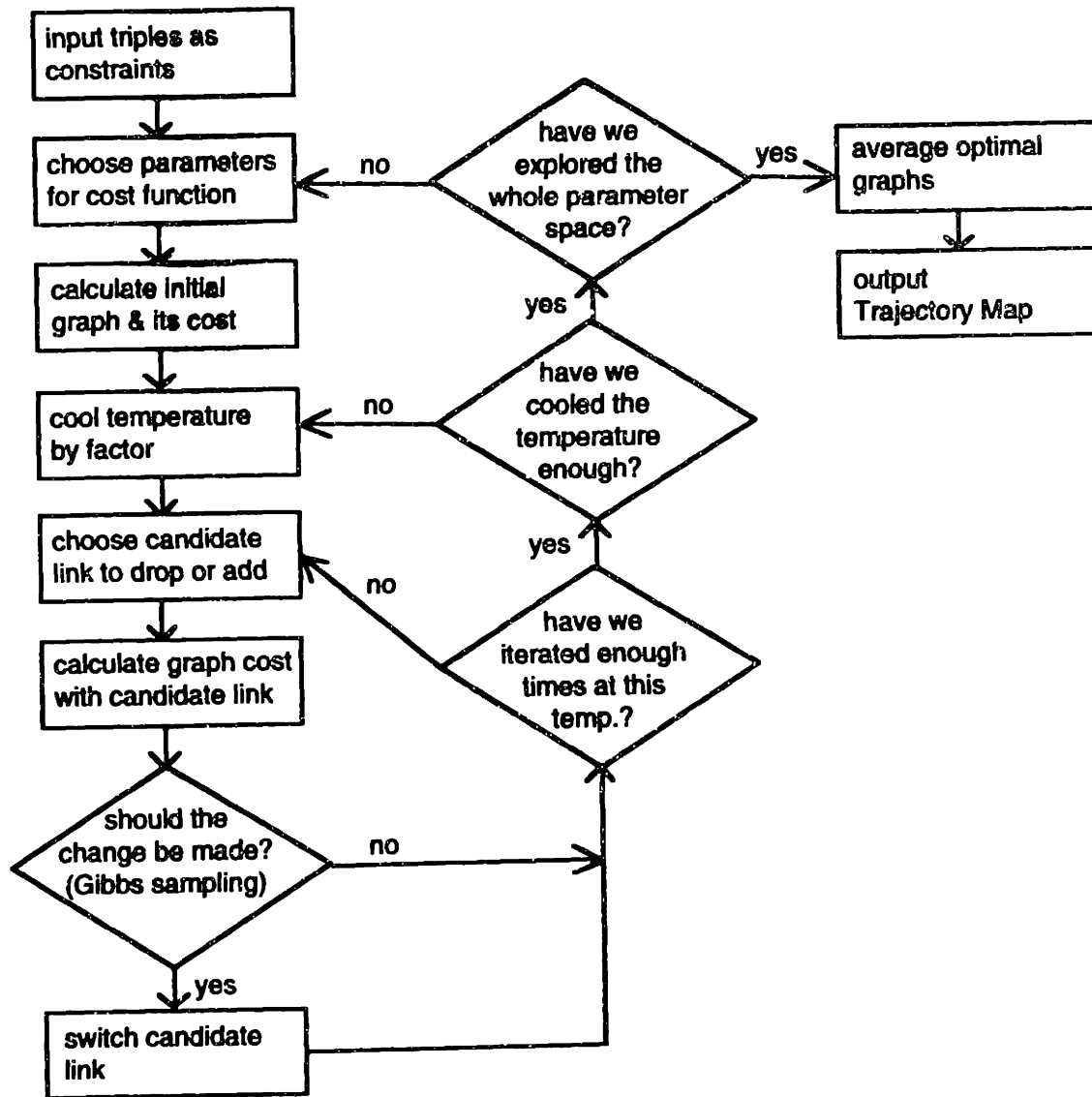


Figure 3: Overview of the simulated annealing portion of the TM algorithm.

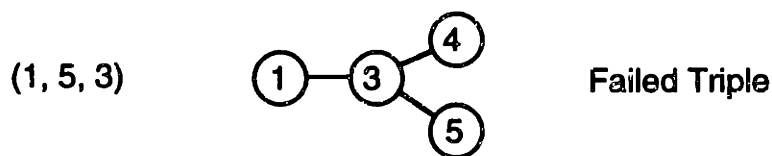
## The Cost Function

The cost function is a mathematical model of our choice of how to settle the various ambiguities that may arise when trying to link triples as described above (Figure 1). We noted the split-or-fit issue and the issue of conflicting triples. Our cost function also represents an attempt to model the manual heuristic of linking triples that Richards and Koenderink (1995) suggest.

The cost function contains five costs, each of which is based on an assumption about the constraining triples. To summarize the cost function, graphs can differ depending on whether the cost parameters emphasize satisfying all the triples (giving a more connected graph) or keeping the graph structure simple (giving a graph with more linear chains and fewer satisfied triples). To break down this issue, we have costs of three types, constraint costs, metric costs, and topological costs.

### Constraint Costs

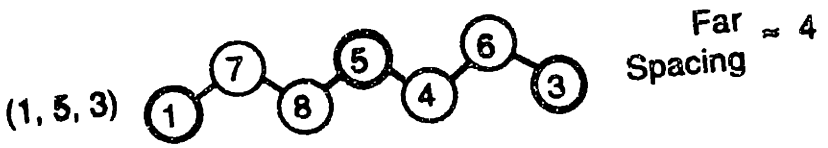
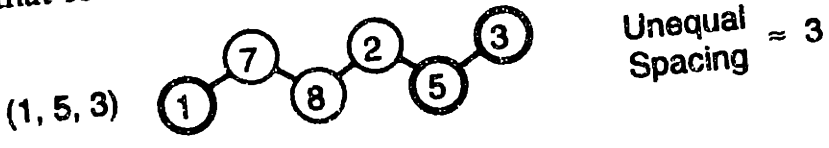
A constraint cost is based on the degree to which the constraints in this optimization problem (the triples) are satisfied. For example, if we need to satisfy the triple 1 - 5 - 3 and the current graph is the one shown below, then because 3 is closer to 1 than 5, that triple is not satisfied. We have one term in the cost function, called the FailedTriple cost, to penalize for unsatisfied constraints. Because we want to emphasize the priority of satisfying triples with higher weights, the FailedTriple cost is the sum of the logs of the weights of the triples that the graph left unsatisfied. This cost stems from the initial idea that the graph should be a good representation of the triples from the subject data.



### Metric Costs

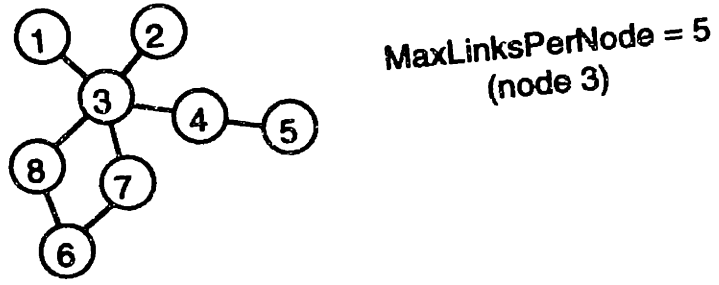
There are two terms to penalize for metric costs, the UnequalSpacing cost and the FarSpacing cost. We suggest the graph incurs metric costs if it includes a triple within the graph, but it does not allow the nodes of the triple to be direct neighbors. If the nodes of a triple are unequally spaced or spaced quite far apart, e.g. 1 - 5 - 3 is satisfied, but the graph distance between 1 and 5 is 1, while the graph distance between 5 and 3 is 4, then the graph has incurred an UnequalSpacing cost. If the nodes are spaced far apart, but are equally spaced, then the graph has incurred a FarSpacing cost (see below). The UnequalSpacing cost for a given triple is the difference between the number of nodes between the three triple nodes, multiplied by the weight of the triple.

The FarSpacing cost for a given triple is the number of extra nodes between the triple nodes, again multiplied by the weight of the triple. These metric costs stem from our assumption in TM that when the subject performs the extrapolations and interpolation in the original quintuplet, she would prefer to pick stimuli that result in close, equidistant quintuplets.



**Topological Costs**

There are also two terms to penalize for topological costs within a given graph. The TotalLinks cost is equivalent to a "price" per link; each additional link has a cost. The MaxLinksPerNode cost encourages links to be spread across nodes instead of stemming from just one or two individual nodes by assigning a penalty proportionate to the greatest number of links on any one node (see below). The TotalLinks cost come from the modeling assumption that the simplest graph possible should be used to model the data (Ockham's Razor). The MaxLinksPerNode cost stems from the assumption that it would be rare for one stimulus within a domain to have many more features associated with it than the other stimuli.



Thus, the cost function can be expressed as:

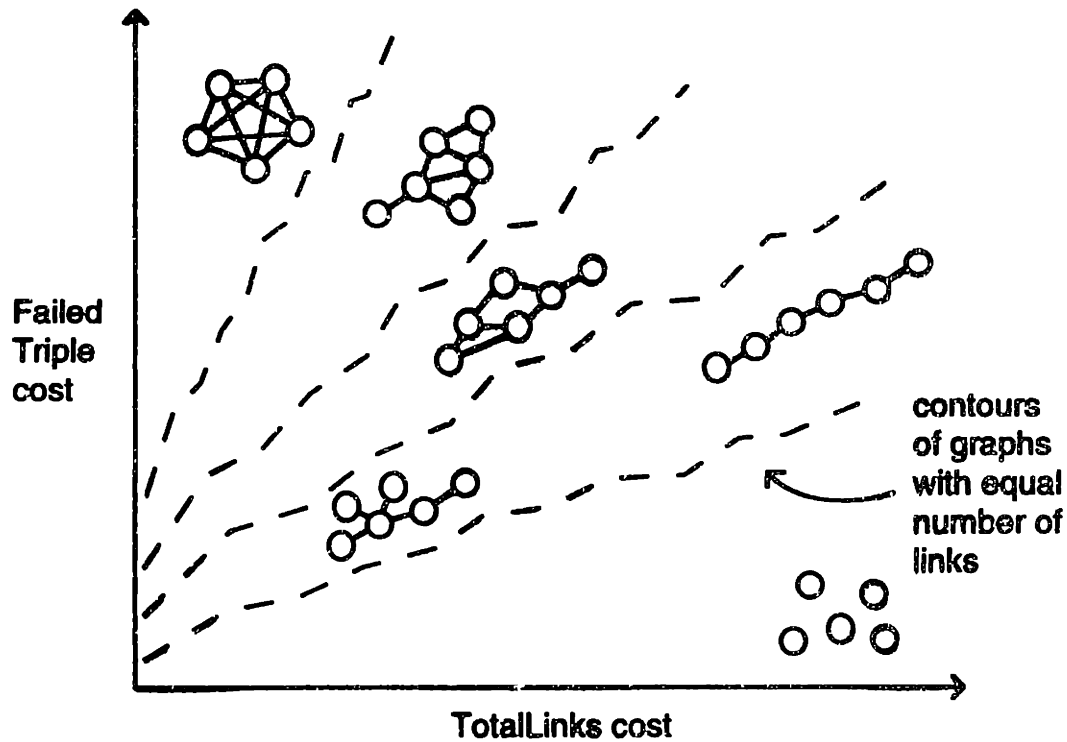
$$cost = w_f x_f + w_{us} x_{us} + w_{fs} x_{fs} + w_d x_d + w_{ml} x_{ml}$$

where

constraint cost	$x_f$ = FailedTriple cost
metric cost	{ $x_{us}$ = UnequalSpacing cost $x_{fs}$ = FarSpacing cost
topological cost	{ $x_d$ = TotalLinkscost $x_{ml}$ = MaxLinksPerNode cost

### The Parameters of the Cost Function

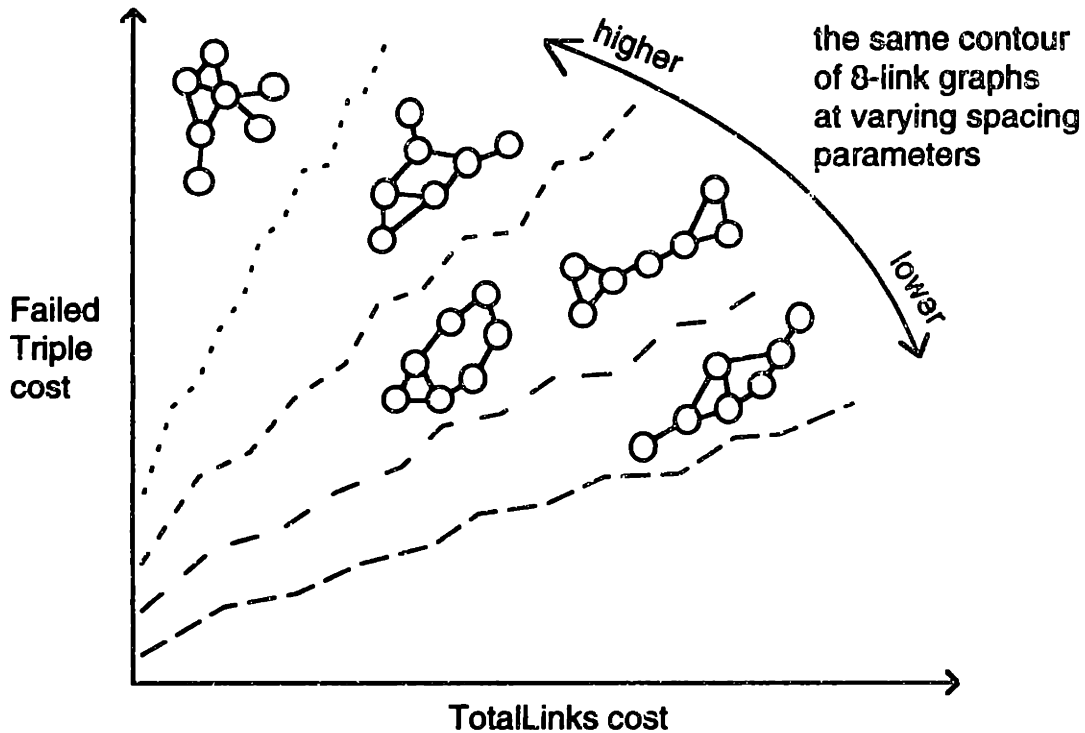
Each of the five terms includes a parameter  $w$  which weights that term in relation to the others. Each vector of parameters  $\vec{w}$  defines a graph space in which we can perform simulated annealing. A good way to think about the parameters is as a set of priorities about whether to satisfy triples or cut down on links. The fewer links there are, the fewer satisfied triples there can be. For example, if the FailedTriple parameter is very low (allowing triples to fail pell-mell) or if the TotalLink parameter is very high (reducing the number of links drastically), then the graph will likely have very few links at all, let alone chains that could make meaningful trajectories. Likewise, if the FailedTriple parameter is very high or the TotalLink parameter is very low, then links will flourish and the graph will be a dense, fully-connected mess. Figure 4 schematically illustrates some of the different optimal graphs one might get by varying the FailedTriple and TotalLink parameters while holding the other parameters constant.



**Figure 4:** A schematic illustration of the trade-off between satisfying triples and keeping graphs simple. The plot shows hypothetical graphs generated from various points in FailedTriple/TotalLinks plane of the parameter space (while holding other parameters constant). The dotted lines mark contours of graphs with equal number of links.

The two spacing parameters, UnequalSpacing and FarSpacing, generally correlate strongly; it is often the case that triples that are badly spaced are also spaced too far and vice-versa. They also tend to cancel each other out, i.e. if one of the spacing parameters is high when the other is low, the overall spacing penalty will be somewhat similar to if they had both been set to the average. Therefore when exploring the graph structure we usually adjust just FarSpacing, and leave UnequalSpacing alone.

When the spacing parameters are low, the graphs tend to stretch out, becoming entirely linear if allowable. When spacing parameters are high, the graph will bunch up, forcing the members of no triple to be more than one link apart. Thus, the spacing parameters have the effect of shifting the contour lines of equal-link graphs radially about the origin of the FailedTriple-TotalLinks origin (Figure 5).



**Figure 5:** The same contour line of 8-link graphs in FailedTriple-TotalLink space under different spacing parameters. The differences in the graphs are subtle; note that internode distances are overall higher in graphs with lower spacing parameters (sometimes 4) than in graphs with higher spacing parameters (mostly 2 with some 3).

Because most of our subject data so far leads to graphs that are already well-distributed in terms of density, the MaxLinkPerNode cost does not assist us greatly in finding optimal graphs. Therefore we will not discuss the effect of the MaxLinksPerNode parameter on the parameter space at this time.

Thus we can cover a useful area of the parameter space by exploring a three-dimensional subspace of the five-dimensional parameter space by using just the three parameters for FailedTriples, TotalLinks, and FarSpacing. Because graphs can vary so dramatically in this space, the algorithm runs at a wide range of parameter settings and then averages the resulting graphs. "Average" means that if link 1 - 3 occurred in 50% of the graphs in the sampled parameter space, then the output graph has a weight of 5.0 (of 10.0) on link 1 - 3.

The question remains as to how to determine this range of settings. The current sampling of settings has been carefully chosen through trial and error on a variety of different types of data. We run the algorithm over a sufficiently wide area of space that we are assured that it reaches extremes in all directions. The extrema can be recognized when graphs are fully connected, or when there are no connections at all. The sampling is not completely uniform; at lower values of the parameters we sample more densely, because smaller changes in that range have a significant effect on graphs. We believe that the ideal subspace within the parameter space varies slightly depending on various attributes of the stimulus domain, but our experiments in that area are not yet finished. Even if the ideal subspace were found, results would not change dramatically; all the stronger trajectories would emerge the same. More subtle paths might be able to be identified more accurately, however.

Once we have a graph, the experimenter can iteratively threshold the graph (remove links below a certain weight) to see how the graph breaks into trajectories. Because the graphs that appear through gradually increasing the threshold approximate the graphs that one finds in the weight space as one moves from the areas of very dense graphs to the areas of less dense graphs, the TM output graph can be seen as a single representation of a set of the graphs in the parameter space.

Our algorithm outputs the graph as a text file that is structured to be read by the Dot and Neato graph-drawing software of Bell Labs (Koutsofios & North, 1993; North, 1992).

### **PART III: Diagnostic measures**

As with any data collection and analysis procedure, diagnostic measures are important for answering questions like, "How well does my model fit the data?", "How noisy is the data to begin with?", and "How similar are these two subjects' models?" Below we describe three diagnostic measures created for TM. The first is a simple test for whether a set of triples is random or not. The second measures the explanatory power of the resulting trajectory map. The third provides a method of comparing different trajectories maps of the same data.

#### **Measure of Randomness**

It is helpful to have a measure of randomness of the subject data. We measure randomness by comparing the distribution of the triple weights from the subject's data with the distribution of weights that would occur if data were created arbitrarily, i.e. created by hundreds of Monte Carlo simulations of subjects.. If the two distributions differ significantly according to a chi-squared test, then we can conclude that the subject's data is worth examining. In passing we note that in a Monte Carlo subject, even with medium-sized N

( $N=15$ ), the likelihood of generating a triple with weight greater than 4 is very small ( $< 2.4 \times 10^{-5}$ ), so any subject data that we find with even one triple with a weight of 4 or greater is already likely to be far from random, as we see in Figure 6.

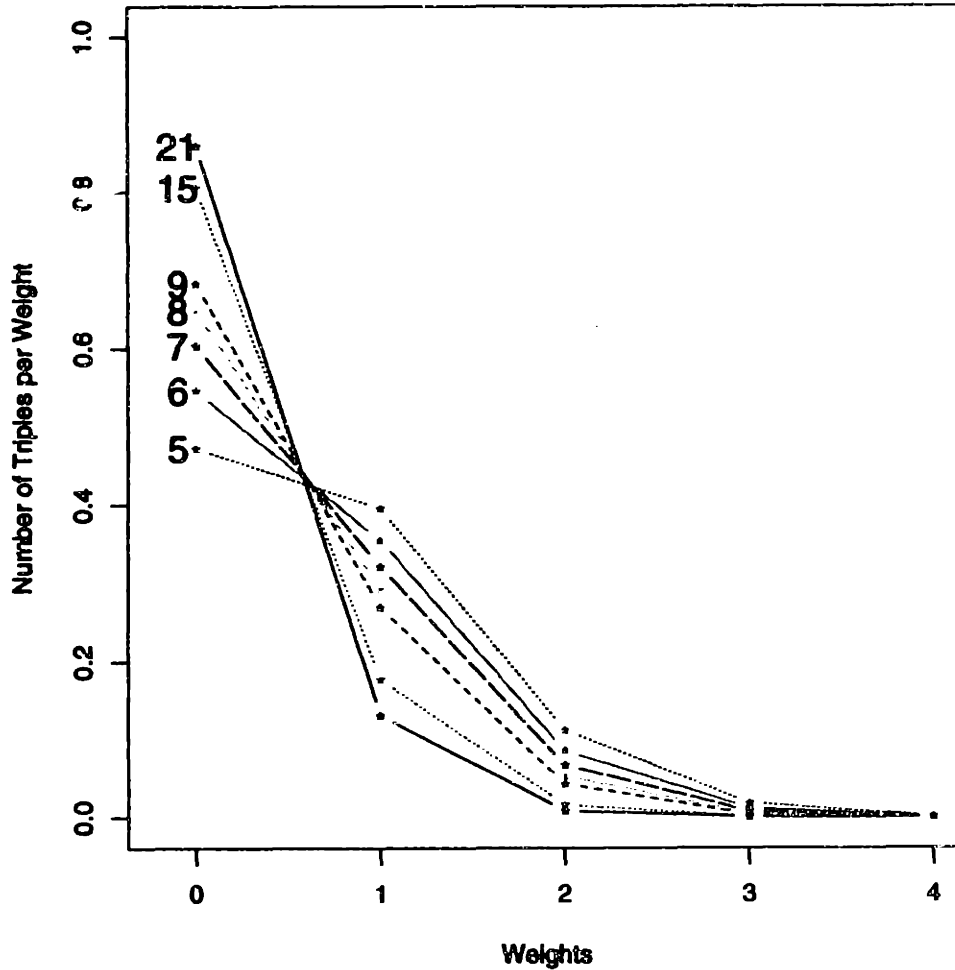
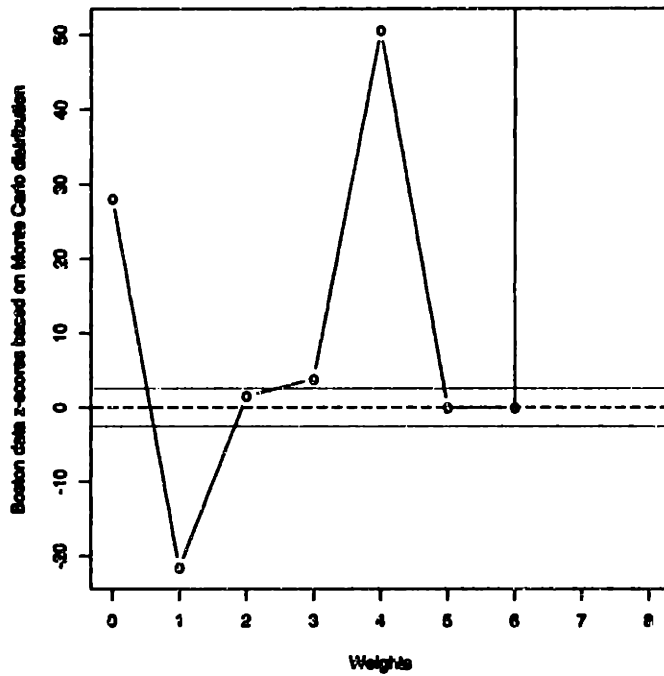
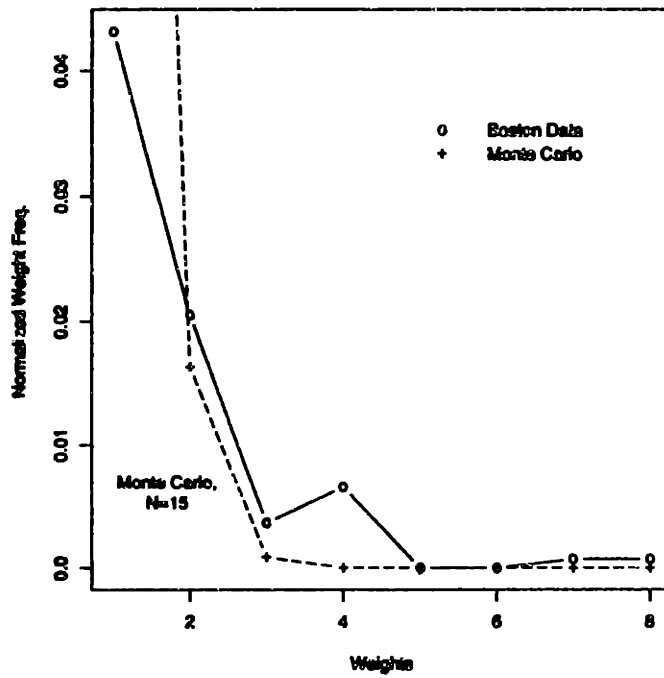


Figure 6: The distribution of triples by weight for various numbers of stimuli ( $N$ ). Each distribution is the average Monte Carlo simulations of either 200 and 400 subjects (larger subject set for  $N > 9$ ).

In Figure 7 we illustrate the power of the Monte Carlo simulations as a test for "noisy" data. If a TM subject ignored our instructions and answered randomly, the weight distribution of the triples derived from his data would match the Monte Carlo distribution closely; all z-scores would likely fall within the dotted lines in the bottom plot of Figure 7, which represent the two-tailed 99% confidence interval ( $z = 2.58$ ).





**Figure 7:** A comparison of data from the Boston tourist attractions data set (15 stimuli) with the Monte Carlo distribution for 15 stimuli (summed from 400 simulations). The figure on the bottom shows the z-scores derived from the difference between each data point and each point in the Monte Carlo distribution (top). These smaller differences are actually the most significant, since variance of the Monte Carlo distribution means decreases dramatically with higher weights.

### Measure of Trajectory Map Fit to the Data

Once we are satisfied with input data, we need a measure of fit, a number which measures how well the output of the TM algorithm models the data. To calculate measure of fit between a set of triples from a subject and the resulting trajectory map, we make a list of the triples held within the model and compare the two lists. See Figure 8. First, to generate a simple trajectory map for this example, links are weighted with the maximum of the weights of the triples that include them. For example, as illustrated in Panel B of Figure 8, link 3-5 has weight 3 because triple 1 3 5 has weight 3 (even though triple 2 3 5, which also passes through 3-5, only has a weight of 2). To assign weights to the model-based triples, costs that are inversely proportional to the link weights are assigned to the links. See the [bracketed values] in Panel B. Panel C lists the triples contained in the graph with their costs. The cost of a triple consists of the cost of all links traversed while moving from the first node to the third node. Once the model-triples (Panel D) are ordered, we can compare this list of triples with the original data (Panel A).

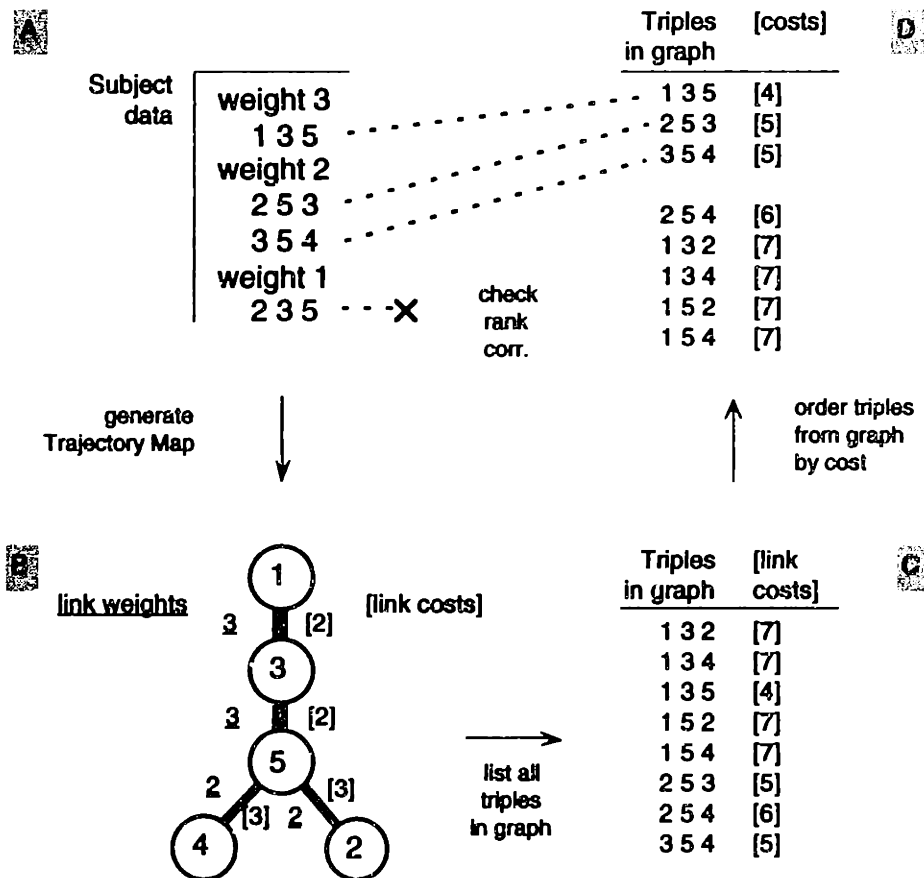
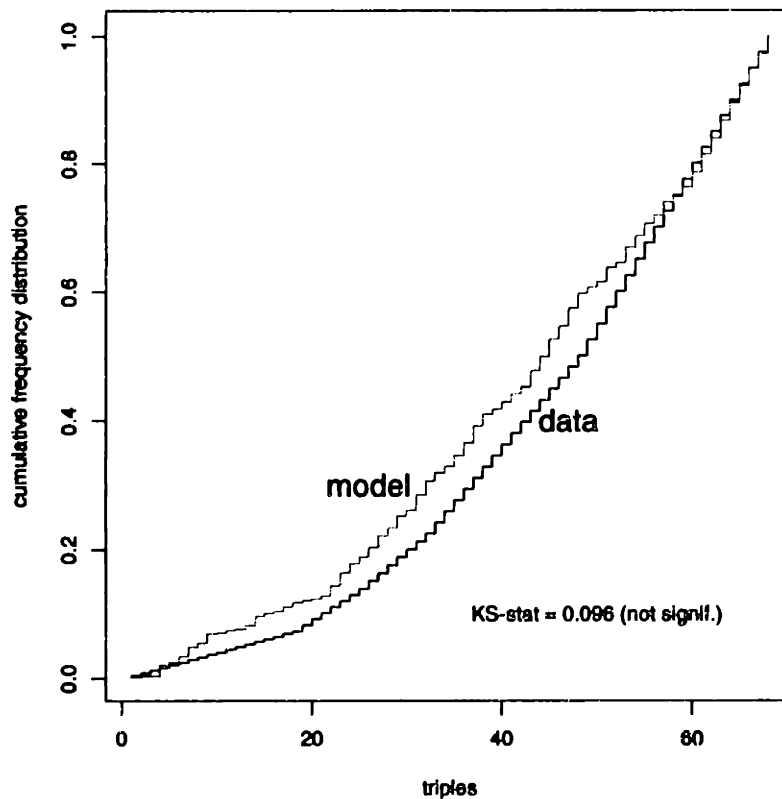


Figure 8: To measure the fitness of graph to its data, we assign costs to each link that are inversely proportional to the weights (Panel B). We note the triples that are present in the graph and their costs (Panel C). Our measures of fit are based on comparing the rank ordered lists of model-triples (Panel D) and data-triples (Panel A).

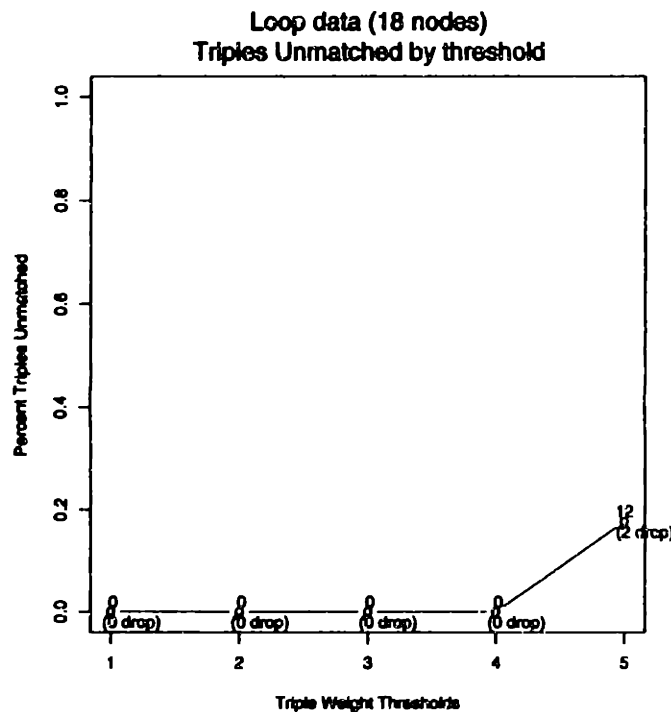
Two different measures are used to assess fit, each based on a comparison of these two lists of triples. The first is simply the percentage of unmatched triples, i.e. the percentage of the data triples that were not included in the model. In Figure 8 (Panel A), this measure is 25%, since the model contains 3 out of the 4 data triples (triple 2 3 5 remained unexplained). This measure does not penalize the model for containing additional triples beyond the data, however, and thus a fully-connected model would satisfy 100% of the triples while offering little insight into the domain.

The second measure of fit is based on ranking the triples in the two lists, and then calculating the Kolmogorov-Smirnov statistic,  $D$ , (Press et al, 1988; Siegel, 1956) for the two cumulative distributions of the two lists of ranks.  $D$  equals the maximum difference between the two distributions (Figure 9). If the model contained exactly the same triples as the data, weighted in the same order, the statistic would be zero. As the model adds additional triples (as it often does just because of the necessary topology of a graph that models other triples), the distribution of model-triple ranks becomes distorted in comparison to the ranks of the data-triples. Thus, this measure penalizes for the additional triples in a model that the matched-triple measure does not take into account. We hope in general that the statistic shows that the two distributions are not significantly different, though even if they are, we have a normalized measure of difference that can be compared across domain.

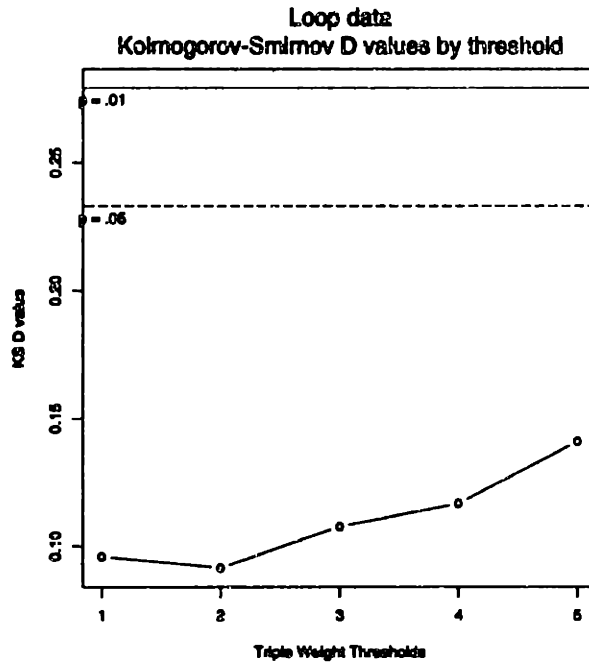


**Figure 9:** A plot of the cumulative distribution functions of the rank orderings of the two lists of triples. The KS-statistic shows that the distributions of model-triples and data-triples are not significantly different. The data are from the Loop data set, Chapter 4).

Both of these measures are key for determining the level of “noise” in the data. Because it is likely that lower-weighted triples contain more noise, triples weighted below a certain threshold are removed before using them as input for the algorithm. The measures of fit are used to determine the threshold. The algorithm is run on sets of triples based on all possible thresholds, and the two measures of fit are calculated for each of the resulting models. Note that the measure the number of matched triples is based on a comparison of the model-triples with *all* data triples, even if the model was based on only triples of weight 3 and above, etc. Figure 10 shows a plot of the percentage of unmatched triples for models that have been created from different subsets of the data-triples, each subset thresholded at a different value. The number “dropped” indicates the number of nodes that are dropped from the model. The model from data thresholded at 4, for example only includes 16 of the 18 nodes.



**Figure 10:** Plot of unmatched triples for models created from data triples thresholded at various values. Number “dropped” refers to the number of nodes dropped from the trajectory map at each threshold. The number above each point is the raw number of triples unmatched. This plot is for the Loop data set.



**Figure 11:** Values of the KS-statistic for the models from data thresholded at different values. This plot is also for the Loop data set.

Figure 11 shows a plot of the KS-statistic (D) for the TM models created by the algorithm at each threshold. The  $p = 0.01$  and  $p = 0.05$  significance levels of D are shown as well, illustrating that our D values are not significant. (A larger D means a greater difference in distributions and a more significant difference.) In other words, the algorithmic model can be considered to adequately account for the triples.

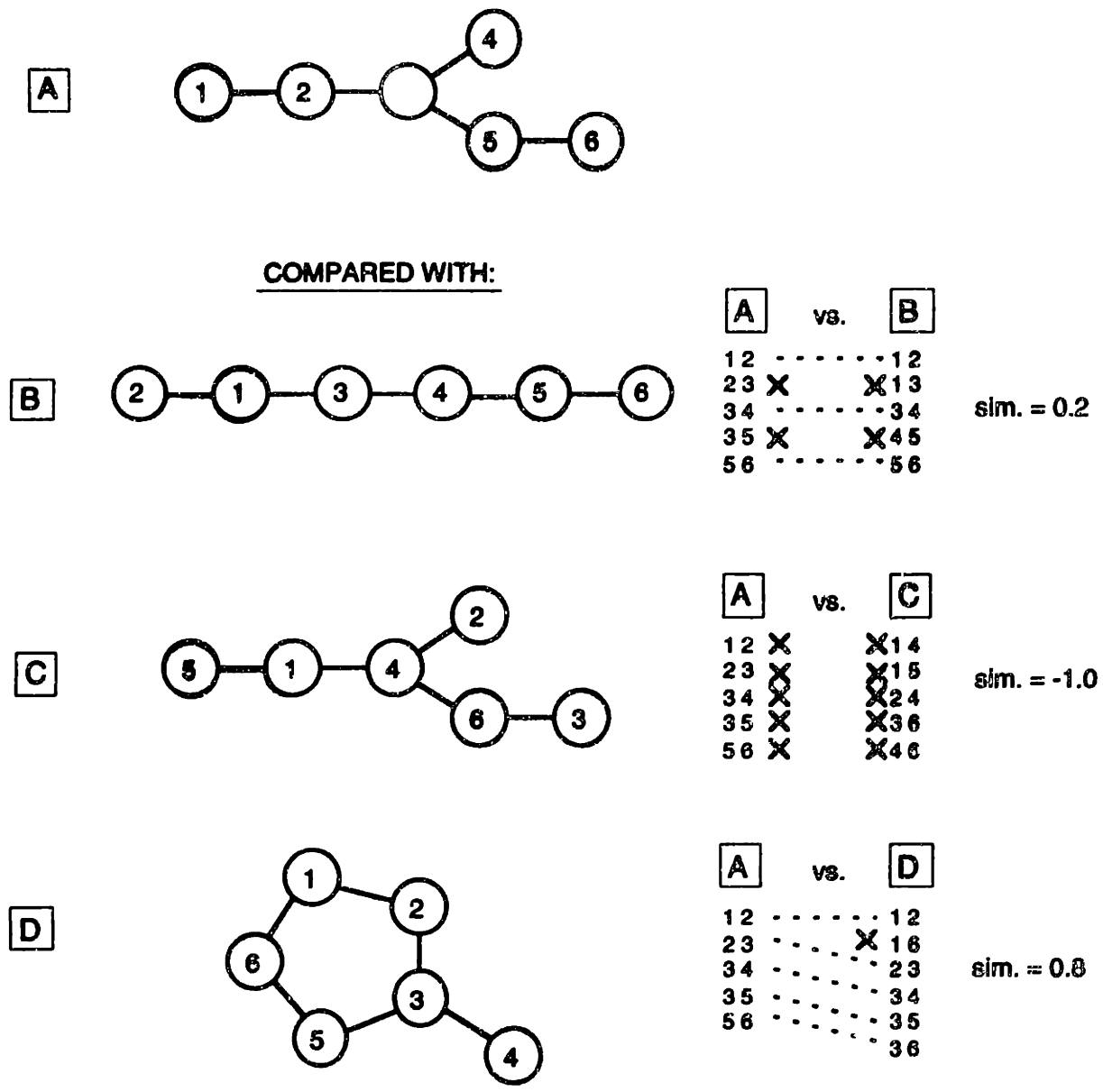
Using both plots (Figure 10 and Figure 11), we can see that the TM model at threshold = 4 would be the best model. It ignores the most triples (assumed noise) without increasing the number of unmatched triples, and the D value at 4 (although higher than at lower thresholds) is not significantly higher. If the D value suddenly passed the significance threshold at 4, we would retreat to a threshold of 3.

### Measure of Similarity of Two Trajectory Maps

Because we have proposed a new model for representing subject data, it is important that we also propose a measure for rating the similarity of two different models. Given two trajectory maps, we look at the lists of the weighted links for both graphs, and reward for common links and punish for distinctive links. To differentiate between graphs with identical topology but differently weighted links, we also penalize for the difference between weights on the common links. The range of this measure is [-1.0, 1.0], where 1.0 implies identical graphs.

$$\frac{(\sum \text{common link weights} - \sum \text{distinct link weights} - \sum \text{diff. in common link weights})}{\sum \text{all weights}}$$

Using such a measure, we can compare the trajectory maps of two subjects and decide whether they might be using similar features to construct their maps. It is important to note that our feature measure focuses explicitly on individual links, as opposed to overall graph structure. In Figure 12 we compare a set of unit-link graphs to give an intuition for the measure. Graph A has a similar overall topology to Graph C, for example, but because their links are actually completely different, they have a similarity value of -1.0. Graph D, on the other hand, has a very different topological structure, but because it is identical to Graph A with the exception of one link, A and D have a similarity of 0.8. Graph B is a version of Graph A that has "absorbed" node 4 into its chain and reversed the order of two of its nodes. The similarity of A and B is only 0.2.



**Figure 12:** Note that the similarity measure is based on common and distinctive links, and not on the overall graph structure; A and D have similar links, for example, while one is a branching tree and the other contains a cycle.

**Summary**

The TM algorithm described above was designed to build trajectory maps from subject data objectively. Based on the simulated annealing, the algorithm uses triples derived from subject data as constraints that can be used to find an optimal connected graph. Cost function parameters were chosen so that the algorithm models the manual heuristics followed by Richards & Koenderink (1995). Lastly, three diagnostic measures for trajectory maps were described: a

measure of subject data noise, a measure of fit for a trajectory map its data, and a measure of similarity between two trajectory maps. We suggest that this algorithm offers an useful method of creating trajectory maps that closely mimics the original intentions of Richards & Koenderink.

### References

- Koutsofios, E. & North, S. (1993) *Drawing graphs with dot: User's Manual*. Bell Laboratories. <ftp://research.att.com/dist/drawdag/dotdoc.ps.Z>.
- North, S. (1992) *NEATO User's Guide*. Bell Laboratories. <ftp://research.att.com/dist/drawdag/neato.doc.ps.Z>.
- Press, W.H., Flannery, B.P., Teukolsky, S.A., & Vetterling, W.T. (1988) *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge: Cambridge University Press.
- Richards, W. & Koenderink, J. J. (1995) Trajectory mapping: A new non-metric scaling technique. *Perception*, 24, 1315-1331.
- Siegel, S. (1956) *Nonparametric Statistics: For the Behavioral Sciences*. New York: McGraw-Hill.



## Chapter 4

### Testing the Algorithm: Automatic vs. Manual Trajectory Maps

#### Abstract

We compare Trajectory Maps generated with the algorithm with maps constructed using a manual heuristic which the algorithm was designed to imitate. In doing so we illustrate TM results in various domains, and show that the algorithm offers satisfactory results for all data sets, and often clearer solutions than the manual technique for more complex data sets.

#### Introduction

In this chapter we challenge the TM algorithm by comparing its results to Trajectory Maps made by hand in several domains. The comparison not only illustrates the general robustness of the algorithm, but also allows us to describe the more subtle nuances of the algorithm by pointing out the differences between the two types of results. We do not focus on analyzing the data sets themselves; several of the data sets have already been analyzed in other papers. After introducing each data set, we take one subject's data, and provide a trajectory map done by the algorithm and done by hand. We compare using the diagnostic measures described at the end of Chapter 3, providing a measure of fit for each graph and a measure of comparison between the two graphs for each data set.

#### The Manual Heuristic

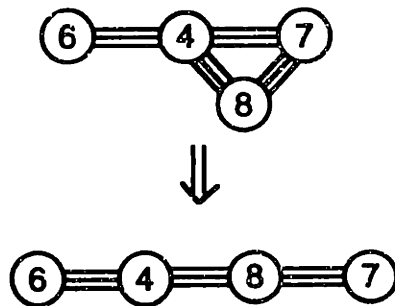
Before continuing to examine particular data, we will briefly describe the manual heuristic of constructing trajectory maps. Note that this construction is always done with arbitrarily numbered nodes, so that the experimenter's manual construction will not be biased by knowledge of the data set. For previously published trajectory maps, MDS scaling and some contextual information were often used to assist construction (e.g. see color data set in this Chapter or visual texture data in Appendix B).

1. Start with the most heavily-weight triples, e.g. triple (1 3 5) with weight 5. Draw the nodes 1, 3, and 5, and connect them with five lines each.



2. Select the next triple of equal weight, if available: draw and connect those nodes. When you find a triple that references nodes that have already been drawn, just append the new nodes to them. Work your way down through the triples, from highest to lowest weights.

3. If you find a triple that cannot easily be satisfied within the graph that you've constructed so far, e.g. (1 5 3), then note that triple as a conflict that needs to be resolved and go on.
4. When you get down to lesser weighted triples (typically below about 1/4 the maximum weight), stop and consider your graph so far. Redraw it so that no links cross, if possible. It's okay to have longer links that curve around so as to avoid crossed links.
5. If you encounter a situation where you can either have 2 short paths between 2 nodes, or can collapse the paths into one slightly longer path, then collapse the paths, esp. if the triples that formed the 2 paths were of equal, high weighting. For example, (6 4 7) and then (4 8 7).



6. Ignore triples of weight 1; they're likely just noise. Add links from weight 2 triples if they help define the structure of the graph usefully or help resolve the conflicting triples that you noted. It is useful to add them with single dotted lines to indicate their weakness.
7. Go over the entire set of triples again, keeping in mind the conflicting triples. If a conflicting triple has equal weight to its "opponent" that is already drawn, do not give preference to the opponent just because it is on paper. Look for indications in other triples (even those of weight 2) that would put a priority on the conflicting triple or on its opponent. You may have to start over completely. Iterate this process several times to make sure that you've satisfied the conflicts as best as possible.
8. Finally, redraw the graph so as to emphasize the heavily-weighted chains/trajectories and to illustrate the overall topology of the graph.

Note that the iterative process of dealing with conflicting triples can become very complicated with a high number of nodes and triples. It is in these cases that the algorithm can become quite useful.

An additional step which can ease the graphic layout of a complex trajectory map is drawing the map over nodes that are positioned in a metric space derived from the adjacencies. As described in Chapter 2, a matrix of proximities can be derived from the number of times that each stimulus appears next to another one in the original quintuplet data. This matrix, called the adjacencies matrix, often correlates strongly

with an independently derived similarity matrix of the data. By plotting the trajectory map over the MDS space from the adjacencies, we can at least usually derive a simple trajectory map with few crossing links (see texture data set in Richards & Koenderink, 1995). We can sometime draw additional inferences from the MDS space as well. This step can obviously be performed with both manual and algorithmic trajectory maps.

## The Data Sets

The following data sets were chosen both to illustrate the abilities of the TM technique and to provide domains with different levels of cognitive abstraction. We first describe a token example domain of black and white circles. We then offer the domain of kinship terms, e.g. "uncle" or "grandmother", a high-level cognitive domain that can be difficult to order. Next, we describe data on colors, a low-level perceptual domain. Finally, we describe data from sound textures, a medium-level domain in that the sounds not only provide the basic aural percept, but also stimulate users to build a model of the source process for each sound. With each data set, we hope to demonstrate the quality of the TM algorithm, and illustrate the features of data that TM can reveal in general.

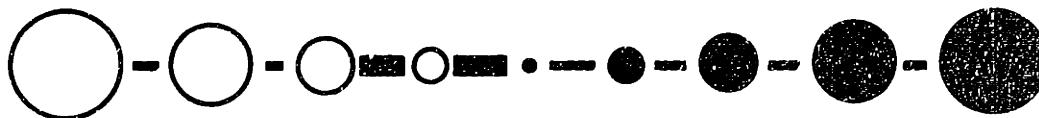
### Circles

This is a data set used largely as an example of our comparison technique. It is the example used as an explanatory example of Trajectory Mapping in Richards & Koenderink. The stimuli are 9 circles of differing sizes and color. 4 are clearly black, 4 are clearly white, and the smallest one could be a tiny circle of either color (because of the black border on the white circles). Figure 1 shows a trajectory map for one subject. By overlaying the subject triples over the trajectory maps, one can divide the map into subgraphs, as shown by the different hatching patterns of the links. No triples cross the small dot in the middle, though trajectories from each side stop at it. This trajectory map is a good example of how trajectories can be seen as ordered clusters. The two clusters are black circles and white circles, and each cluster is ordered by size.



Algorithm solution

Fit: matched 12/12 triples (0% unmatched)



Manual solution

Fit: matched 12/12 triples (0% unmatched)

**Figure 1:** Trajectory maps of circle data. A comparison of the graphs reveals a similarity of 0.93, with 8 common links and 3 more links in the algorithmic solution.

## Discussion

One might wonder why the algorithmic solution contained the 3 weak side links, especially if the rank correlation was higher without them. It is important to remember, at this point, that the rank correlation of the triples, i.e. the order of their weighting according to the graph vs. according to the data, is not of supreme importance. We want the value to be relatively high, meaning that the triples with maximum weight in the data are the most heavily weighted triples in the graph, but we also must acknowledge noise that can arise in the manual from serial construction, i.e. beginning with one triple and then adding others. By satisfying the ordering of the subject's triples very closely (as is done in the manual technique), we actually find a less intuitive link structure.

For example, the manual result contains heavy links between the smaller white circles, and only medium weight links everywhere else. This asymmetry results from that particular triple having a heavier weight than the others. The algorithm generalized over the triples, establishing heavy links throughout the salient structure of the graph.

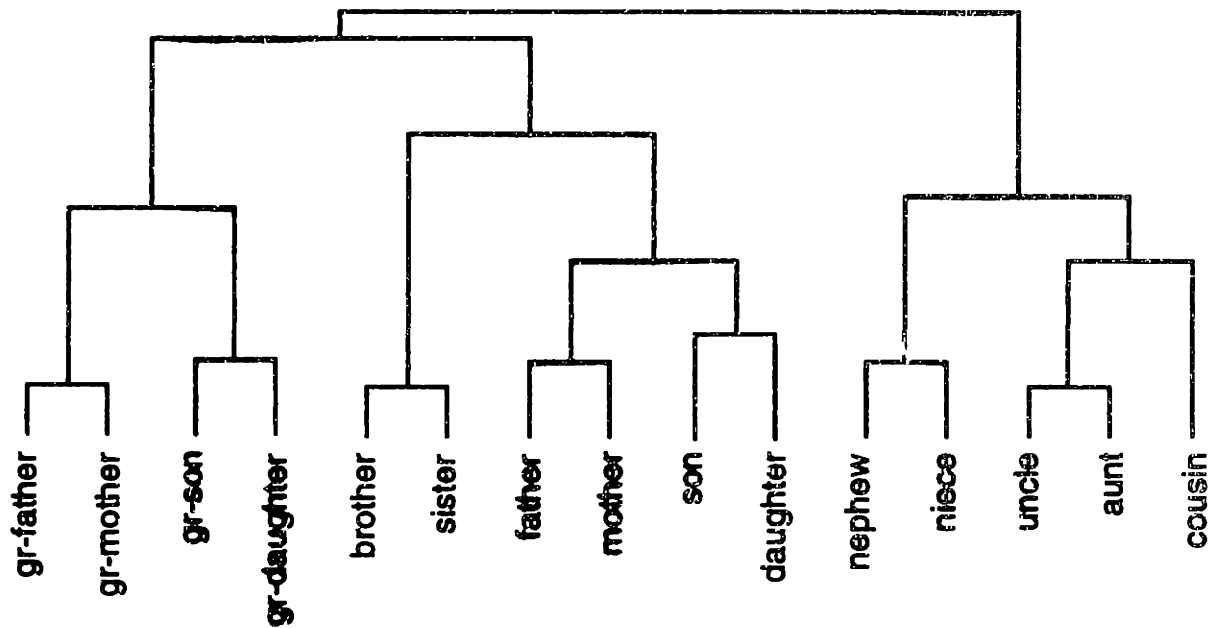
Lastly, one might consider the weak side links of the algorithmic solution to be an indication that subjects are ambivalent about which circle of a given color should be the "middle" circle, given the largest and the 4th largest.

## Kinship Relations

This data set of English kinship terms was originally used in a free-sorting task by a variety of cognitive anthropologists (see Tyler, 1969); it has more recently been analyzed by John Daws using MDS and cluster analysis (1996). The terms are *grandmother, grandfather, mother, father, aunt, uncle, daughter, son, niece, nephew, granddaughter, grandson, and cousin*. We had subjects perform TM with the stimuli as comparison to the clusters that Daws presents.

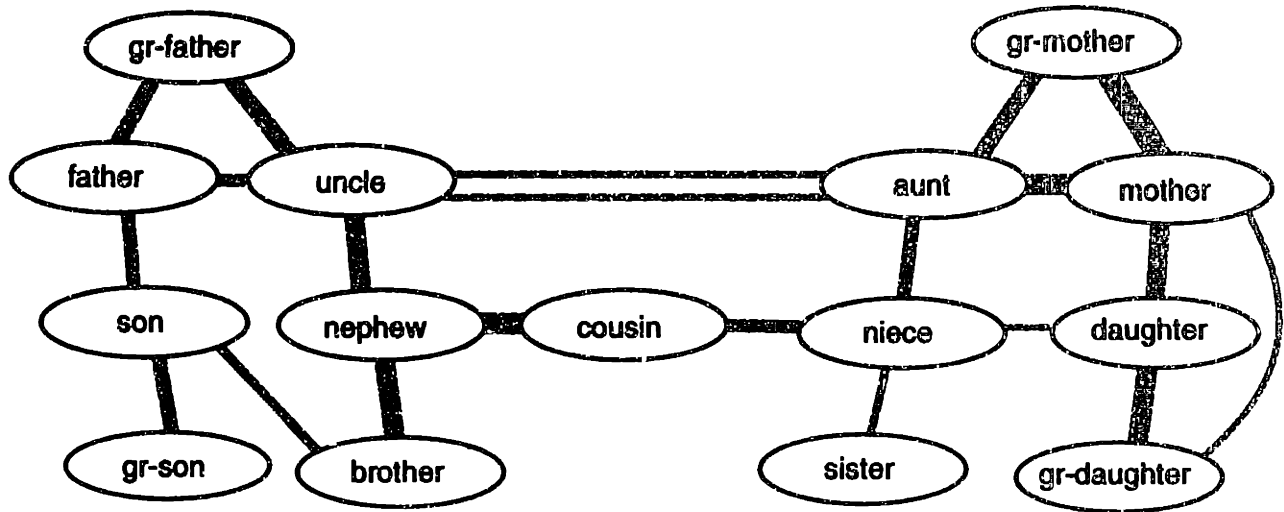
Figure 2 shows a clustering solution based on the original Rosenberg and Kim data, as presented by Daws. Although it could be that Rosenberg & Kim's subjects were using a different measure of similarity than we have, their clustering solutions does not seem conceptually appealing; it represents the features of neither gender nor generation. It seems mainly to group terms by proximity in a typical family tree, with the assumption that two grandparents and two grandchildren are equally close, for example.

In our TM solutions for one subject below (Figure 3), we see how TM is particularly appropriate for this data, since its tendency to form ordered clusters illustrates the two main features of this data set. The genders are clustered, males on the left and females on the right, and the generations are roughly ordered within each cluster. *Cousin*, because it is genderless, spans the gap between the clusters of the two sexes. We predict that languages have only gendered terms for cousin, such as German, would have the cousin terms included in the appropriate clusters.

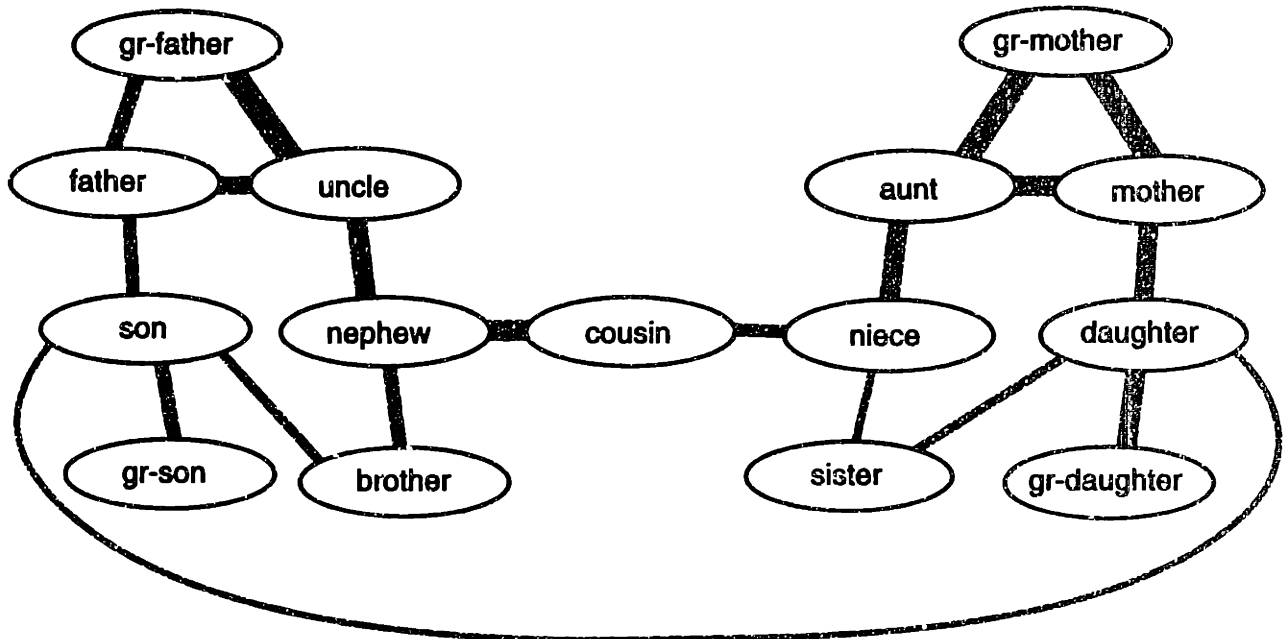


**Figure 2: A hierarchical clustering of kinship terms according to Daws's cluster analysis (1996). Note that this clustering does not explicitly represent gender; each linguistically complementary pair of terms is clustered, but that does not tell us that one member of each pair have something in common. It also does not represent generations. The main features of this tree seem to be based on proximity in a typical family tree.**

**It should be noted that subjects found these stimuli particularly difficult to work with during TM because of the changing reference frames of the terms. One subject reported localizing himself as "son" and performing the entire TM task in that context.**



Algorithm solution  
 Fit: matched 79/113 triples (30% unmatched)



Manual Solution  
 Fit: matched 98/113 triples (13% unmatched)

Figure 3: Trajectory maps of kinship data. A comparison of the graphs reveals a similarity of 0.68, with 16 common links and 4 distinct links on each graph.

### Discussion

Here we see that the graphs fit only about half of the triples in data! Such low numbers indicate a significant number of directly conflicting triples in the subject data. Such conflicts can be easily understood within the context of this data set, since the issue of different reference frames could be confusing for subjects. If

subjects do not choose a consistent way of interpreting the stimuli, then they will likely producing conflicting triples.

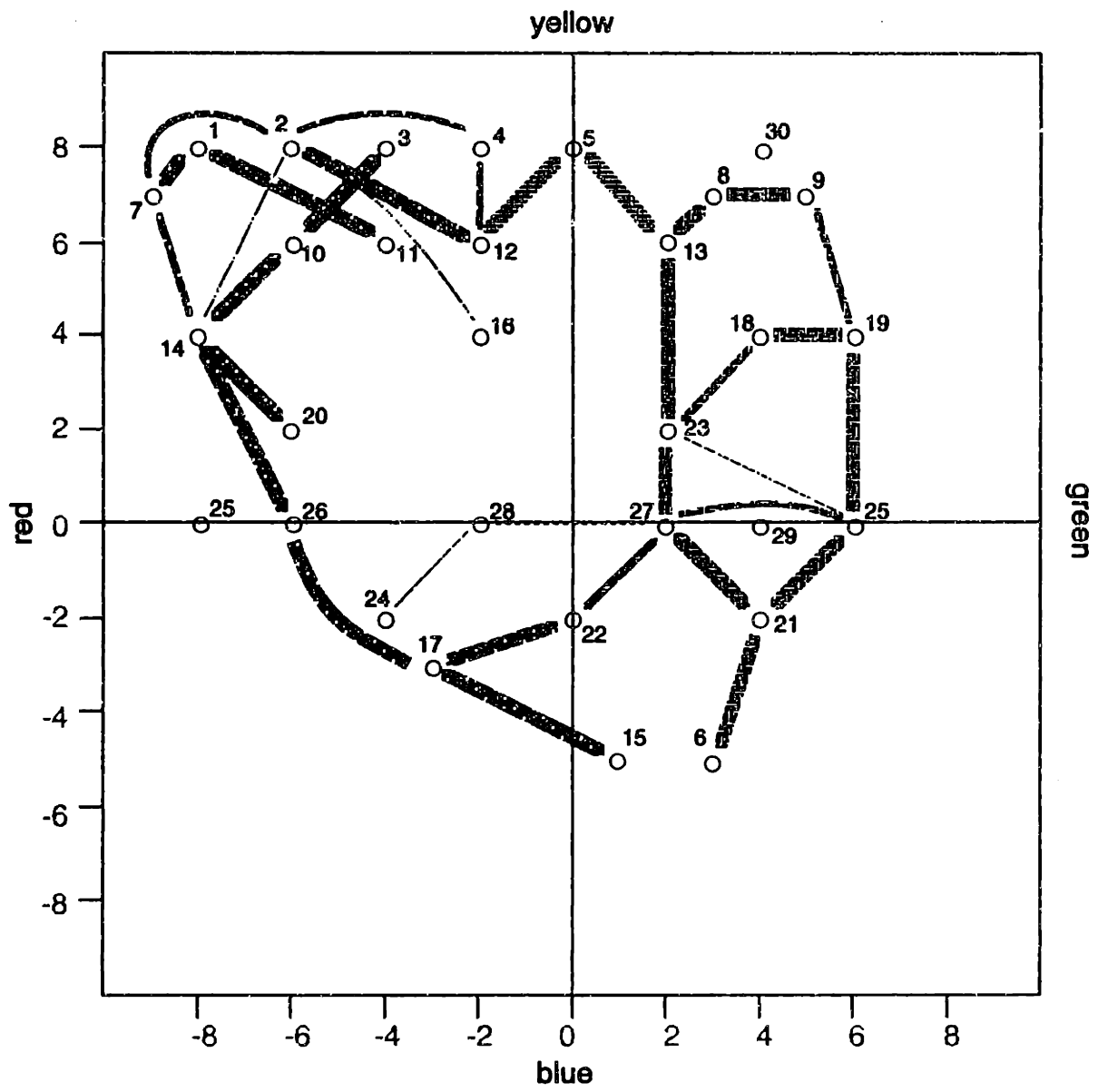
The difference in links is not overwhelming; the most heavily weighted paths of the graphs are similar. Only weaker links differ. It seems that the subject had a bias towards the matriarchy of the family. It is worth noting that the two solutions had the same number of links, but the manual solution satisfied three more triples than the algorithmic solution. This difference is a sign either that the algorithm is still imperfectly modeling the manual procedure, or that the algorithm has produced the result of various local minima. Future work would attempt to pin down the cause of this difference.

### **Colors**

This data set of colors was introduced by Richards and Koenderink (1995). They chose 38 colors from the LJK uniform color space described by the Optical Society of America (OSA). All colors lie on or just adjacent to the equilightness plane where  $L = 0$ . The manual solution below is a close approximation of the solution presented in their paper. The trajectory map solutions (Figure 4 and Figure 5) for pooled subject data have been drawn over the colors as they lie in the LJK uniform color space to provide some intuition into what the paths represent. There are so few triples in this data set (27) because the triples of weight 1 are not available in Richards & Koenderink.

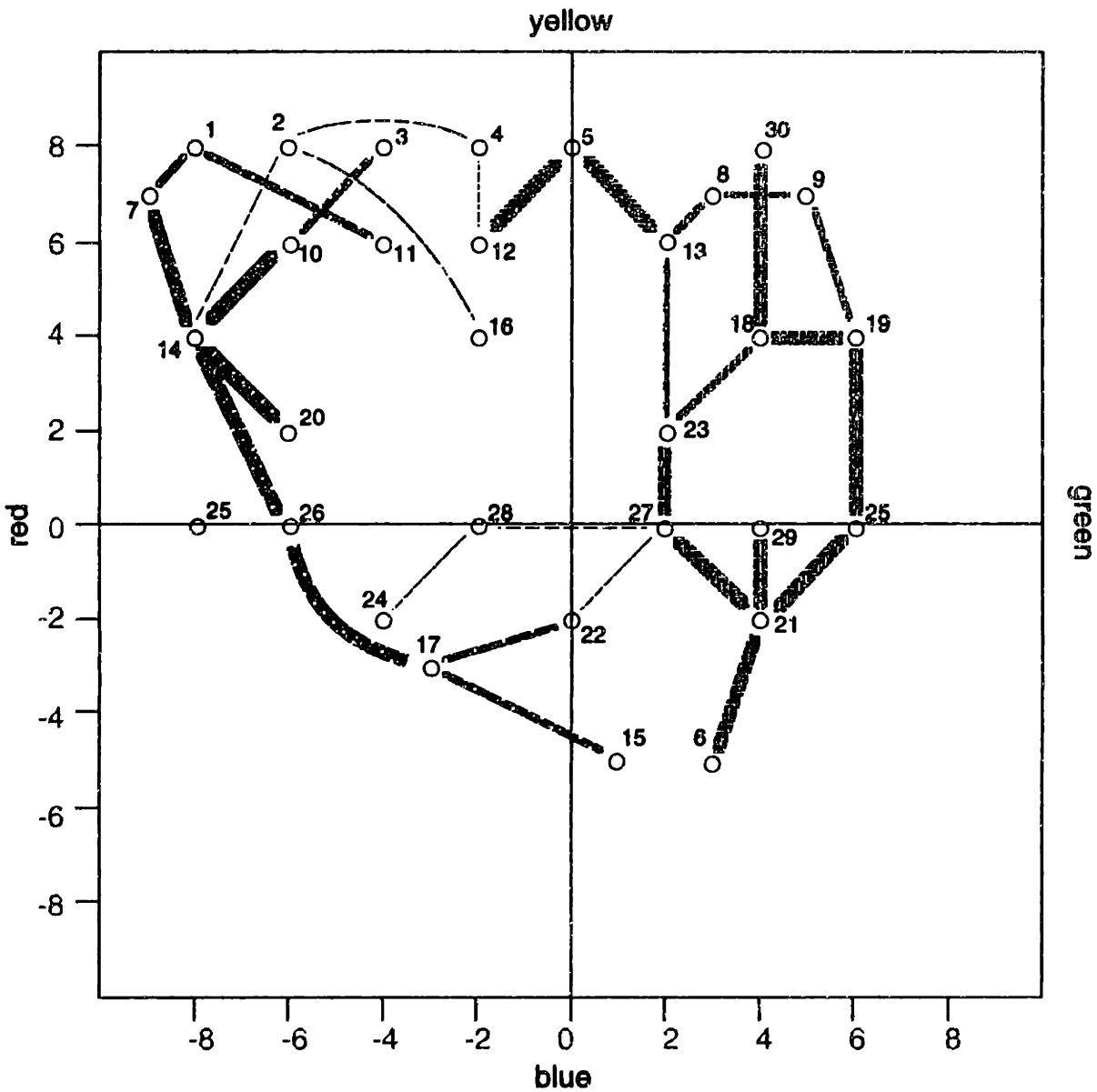
There are three points of interest in this data. First, note that we have divided the graph into three regions that overlap by one node at the boundaries. One region sits mostly in the red half of the plane, bound at nodes 12 and 27; the other region sits in the green half of the plane, bound at 13 and 27. A small subgraph spanning nodes 12, 5, and 13 completes the trajectory map. These regions are based on the layout of the subject's triples over the graph; no triples cross the region boundaries. The regions are a good example of TM's ability to find ordered clusters of stimuli; each region or cluster, though not strictly linear, has an explicit ordering within it that can be used to infer features of the clusters. Note also that TM builds clusters that can overlap; the boundary nodes (12, 13, 27) are all shared by both of their bordering regions.

Secondly, none of the most salient trajectories travel through the color gray, which lies at the origin of the space. The main paths travel around the gray region. We can infer from this fact that the subject would find it conceptually difficult to interpolate between red and green or between yellow and blue, although gray is a fine computational solution. Finally, the fact that the yellow-blue axis seems to divide the space, as opposed to the red-green axis or any other axis presents a research issue worth further exploration. This division is supported by the smallest region that does cross the yellow-blue axis, but does not connect with the other regions.



**Figure 4: Algorithm solution**  
 Fit: matched 18/27 triples (33% unmatched)





**Figure 5: Manual Solution**  
 Fit: matched 18/27 triples (33% unmatched)

A comparison of the graphs reveals a similarity of 0.68, with 31 common links, 2 distinct links on the manual solution, and 4 distinct links on the algorithmic solution.

### Discussion

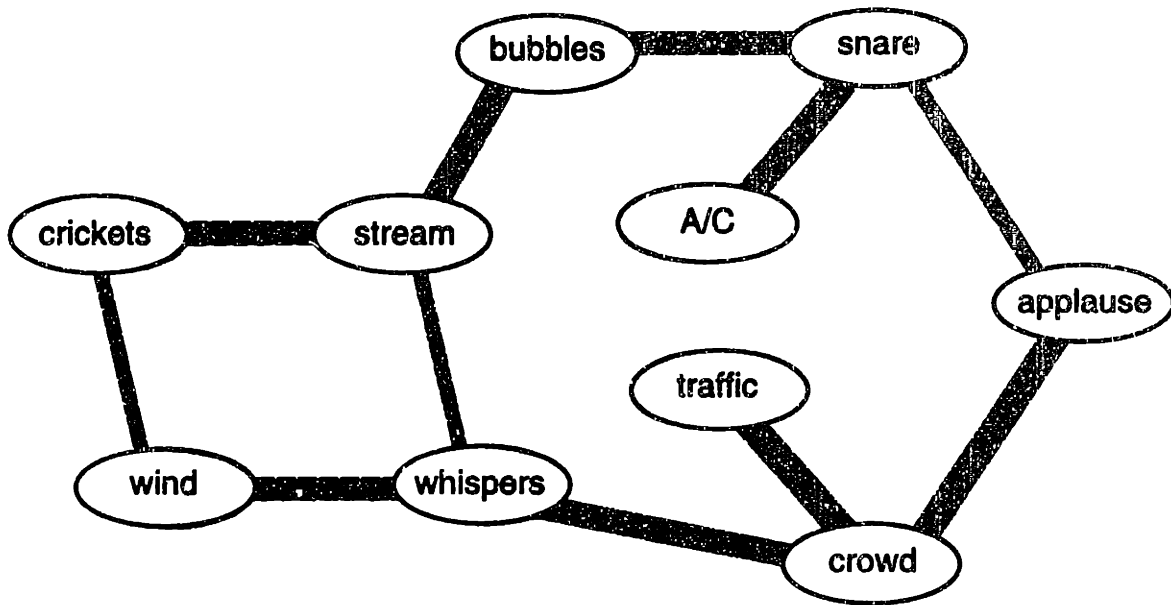
We notice that these two solutions are basically identical, which gives reassurance that the algorithm is useful, since it is particularly difficult to construct the trajectory map of this complex data set by hand. Some of the main differences between the two solutions (e.g. links 18-30 and 21-29 in the manual solution) stem from quintuplets that contained the “not available” response. The current algorithm does not accommodate “not available” and “dead end” responses robustly, whereas an experimenter can accomplish this by hand.

## Sound Textures

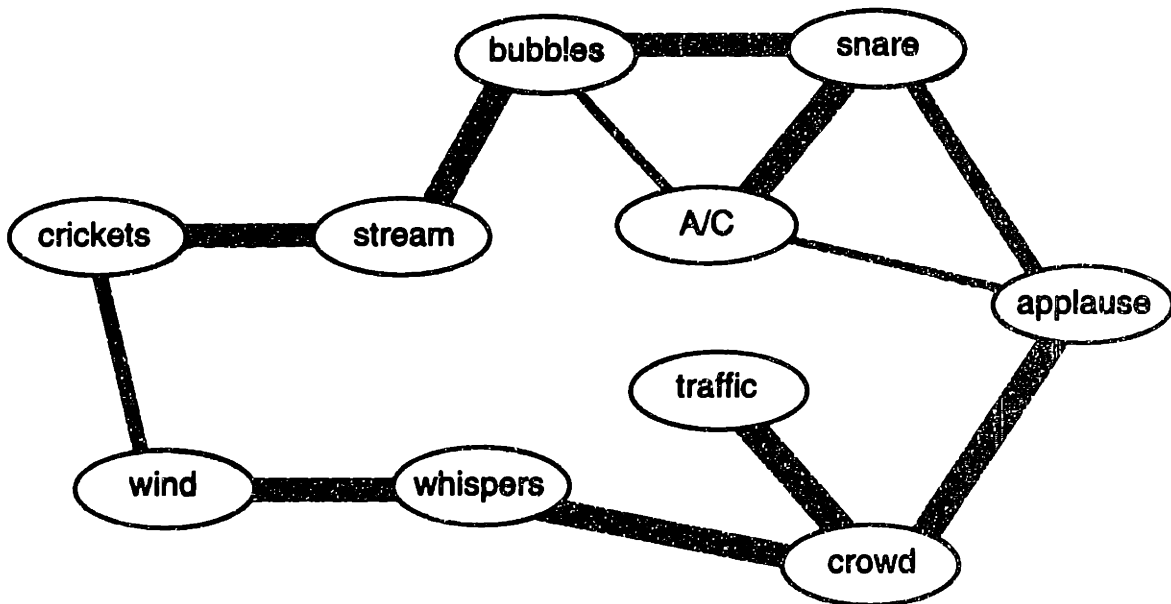
This data set of sound textures was introduced by Nicolas Saint-Arnaud (1995). He defines a sound texture as “an acoustic signal characterized by its long-term properties” (1995, p. 12). Often when people use the word “sounds” they imply short duration sounds that carry a message of some sort, such as a computer’s beep, or a duck’s quack, or a referee’s whistle. Sound textures, on the other hand, are longer duration sounds that usually carry no succinct meaning, such as the sounds of a waterfall, a humming refrigerator, or wind.

The trajectory maps of this data set (Figure 6) come from a subject different than the subject whose data was used in Chapter 2. The maps reveal two main divisions of the stimulus set. The upper region in each map (as drawn here) contains sound textures which are continuous and have distinct pulses of sound that one can distinguish; the nodes within the region change from more distinguishable pulses (*air conditioner motor*) to barely separable pulses (*swarm of crickets*). Although the rhythm of pulses is not always completely regular, this region can roughly be called periodic. The lower region contains a set of aperiodic textures which run from being discrete (*applause*) to continuous (*wind*).

As the algorithmic solution illustrates, the two regions overlap in some areas. The reader may wonder how the boundary textures might be perceived as both periodic and aperiodic. One can imagine this perception most easily with *applause*, perhaps; near the end and beginning of an applause, the sound is more discrete; one can hear individuals. When an entire audience claps, however, the sound texture becomes a continuous sound in which one can hear waves of clapping emerge from the overlap of each individual’s clapping rhythm. Even at this point, however, one can also still hear the claps themselves. A thousand *crickets* give off the same wave-like sound, except it is more difficult to hear individual chirps. The significant overlap can also be explained by the high-dimensionality of the sound texture feature space; because each texture has so many features that the subject could attend to, her perception of a given texture is likely to be significantly influenced by the context of the pair of textures in each trial.



Algorithm Solution  
 Fit: matched 49/68 triples (28% unmatched)



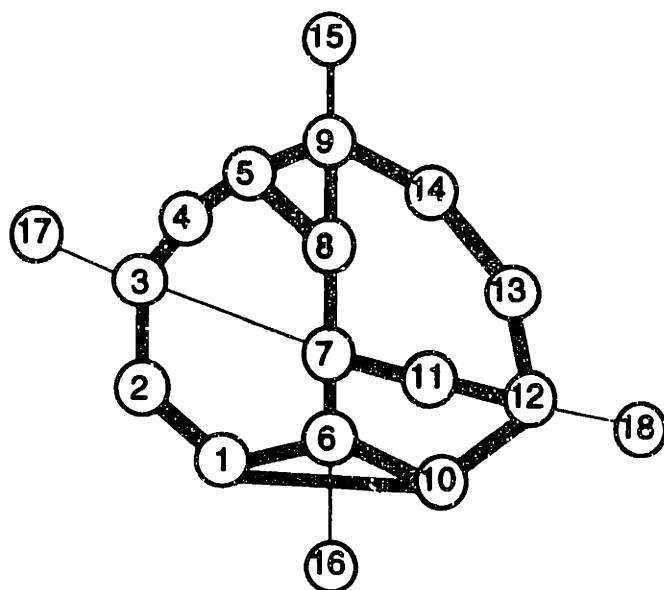
Manual Solution  
 Fit: matched 55/68 triples (19% unmatched)

Figure 6: A comparison of the graphs reveals a similarity of 0.71, with 10 common links, and 1 distinct link on the algorithmic solution, and 2 distinct links on the manual solution.

**Discussion**

Again, as we saw in the case of the kinship data set, many of the triples seem to have conflicts with each other. The only links of difference in our case are the two additional ones in the algorithmic solution: *stream-whispers* and a weak *bubbles-applause*. One explanation for the lack of these links in the manual solution is that





**Figure 8: Manual solution**  
 Fit: matched 148/156 triples (5% unmatched)

A comparison of the graphs reveals a similarity of 0.45, with 18 common links and 3 more links in the algorithmic solution and 5 more in the manual solution.

### Discussion

The main differences between the algorithmic and manual solutions are the position of node 5, and the low weights on the "spur" links. Node 5 is likely in the wrong place because the complexity of this data set makes it difficult to reach the final global minimum solution without stopping at a local minimum. The spur links, i.e. those connecting nodes 15, 16, 17, and 18, are low in weight because the manual heuristic suggests that one ignore triples that have a weight significantly lower than the maximum triple weight. This rule is sensible, since lower-weighted triples are usually due to noise. In this simulation, however, since all triples were consistent, even triples of weight 1 were useful, and they might have assisted the constructor. Also, information about dead ends would have been helpful for the manual construction. Note that the spur nodes will likely be present in at least half as many triples as nodes that have multiple links attached to them, thus, because the lower weighted triples were ignored, triples containing the confirming information about the spur nodes were ignored. The algorithm, on the other hand, takes all triples into account, but can discount their influence on the graph logarithmically.

### Overall Summary & Conclusions

We have introduced five different domains of data, and compared trajectory maps created by the simulated annealing algorithm and through the manual heuristic in each one. The summary statistics (as developed in Chapter 3) are shown below in Table 1. We offer statistics for both data sets in this chapter and some additional data sets discussed in other chapters and in Appendix B. It is worth noting again the subtleties of each measure. The rank correlation is high if the triples in the trajectory map are weighted in the same order as triples in the subject data. The

more important triples measure is the percentage of triples in the subject data that the trajectory map satisfies. Note that most graphs satisfy a high percentage of triples. The number of satisfied triples is unusually low in the representations data set because of the way subject data was pooled, adding noise (see Chapter 5). By examining the data sets for which we have manual constructions, measures show that both methods of constructing trajectory maps can provide a model that represents the data well, and that their results are usually similar.

	algorithm triples matched	manual triples matched	similarity. of maps
Circles	100%	100%	0.42
Kinship Terms	70%	87%	0.68
Colors	67%	67%	0.68
Sound Textures	72%	81%	0.80
Loop Test	99%	95%	0.45
Musical Intervals	98%		
Subway Stations	100%		
Representations	46%		
Boston Sites*	83%		
Visual Textures*	56%		

**Table 1:** Measures of fit for algorithmic solutions and manual solutions, with a measure of similarity of the two for several datasets. Measures of algorithmic fit are provided for other data sets mentioned in the thesis. Those marked with an asterisk (\*) appear in Appendix B.

By achieving similar trajectory maps with the algorithm, we have shown that our model of what assumptions and goals the manual heuristic entails is at least sufficient enough to achieve these results. To review (from Chapter 3), our algorithm model assumes that the trajectory map should 1) satisfy as many triples as possible, emphasizing the higher weights with a log scale, 2) connect the nodes of triples with as short and even a spacing as is possible, and 3) collapse parallel or bifurcating paths when possible so as to maximize simplicity. Because each TM data domain allows us to infer features of the data that MDS or clustering would not necessarily have revealed on their own (see Chapter 2), we recommend TM as a complementary approach to traditional scaling techniques.

## References

- Daws, J.T. (1996) The analysis of free-sorting data: Beyond pairwise cooccurrences. *Journal of Classification*, 13, 57-80.
- Richards, W. & Koenderink, J. J. (1995) Trajectory mapping: A new non-metric scaling technique. *Perception*, 24, 1315-1331.
- Rosenberg, S. & Kim, M.P. (1975) The method of sorting as a data-gathering procedure in multivariate research. *Multivariate Behavioral Research*, 10, 489-502.
- Saint-Arnauld, N. (1995) *Classification of Sound Textures*. Masters Thesis, MIT Media Lab. To receive a copy email docs@mit.edu or call MIT Document Services at 617 253-5650.
- Tyler, S.A. (Ed.) (1969) *Cognitive anthropology*. New York: Holt, Rinehart & Winston, Inc.





## Chapter 5

### A "Meta" Problem: Representational Forms & Functions

#### Abstract

Knowledge access and ease of problem-solving depends on our choice of representation. Because of our unique facility with language and drawings, linguistic and pictorial descriptions are typically taken as the simplest, most abstract categories of representation. We note the importance of mixed representations and offer evidence that when subjects organize representations according to their use, they categorize not just as this pictorial level of characterization, but at multiple levels: computational, functional, and pictorial.

#### Introduction

Our choice of mental representation can make an enormous difference in the way that we analyze a problem, conduct a search, or convey instructions or ideas to a colleague (Winston, 1980; Larkin & Simon, 1987; Moray, 1990; Norman, 1993). We have a large variety of representations available to us, and in this paper we explore how people relate them to each other. Our ability to choose appropriate representations given a problem demonstrates that we employ a mental representation of representations.

Several researchers have proposed theoretical frameworks for classifying representations, often focusing specifically on one aspect of the representations without adequately distinguishing their different characteristics. Bertin (1967), Twyman (1979), and Lohse, et al. (1994), for example, offer frameworks for representations based on their external visual appearance or "physical structure" (Lohse, et al. p. 36). Their frameworks are useful for graphic classification purposes, but do not tell us how people might organize the representations for efficient cognitive use. Cox & Brna (1993), in an effort to explore the different uses between "graphical and linguistic" representations, clustered 87 representations into categories according to both computational and external characteristics, but did not distinguish these levels of analysis.

To clarify the different aspects of representations that one might consider, we suggest the three characteristics shown in Figure 1. We will use the term *representational form* to refer to the computational framework of a representation which offers the infrastructure for carrying out the organizing principle on the content. Some examples of representational forms are list, Euclidean metric, hierarchical clusters, and sets. We use this term to distinguish these computational structures from graphic or *external depictions*, the physical entities that illustrate representational forms, e.g. grid, sketch, or icon. Lastly, one can also characterize a representation by its function (intended or otherwise) and content.

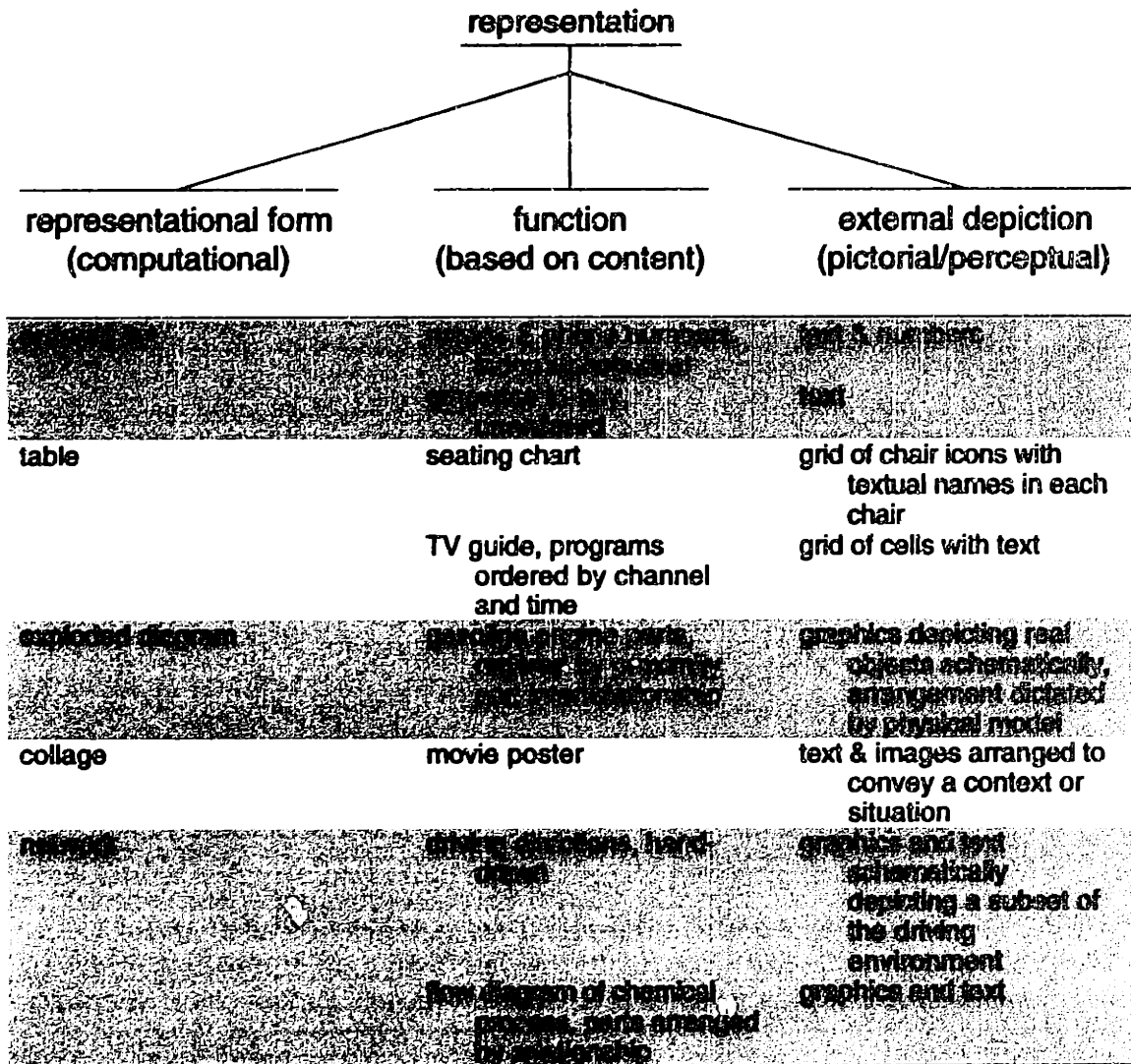


Figure 1: There are three salient aspects of a representation that should be distinguished: its representational form, its function, and its external depiction.

We want to explore which characteristics subjects consider to be salient when organizing representations according to their *use*, as well as what types of features they use within those levels. If we have subjects use a variety of representations, for example, will the subjects later group them by their function, their computational structure, or their external depiction?

To examine this question, we have solicited subjects' judgments about the relationship between 17 different representations. This approach differs from previous work in two ways: 1) we solicit subjects' judgments according to the use of the forms, and 2) we use trajectory mapping (Richards & Koenderink, 1995) as an experimental approach (instead of a scaling technique based on similarity). We use

trajectory mapping (TM) because it forces subjects to think about the specific features shared by forms instead of their general “pictorial” similarity.

## Method

The 17 representations were chosen to be examples of a wide variety of representational forms. To prevent subjects from using pictorial parameters to judge the representations, we presented each representation through a short verbal description. For example, for the representational form “linear ordered sequence of both textual and pictorial elements”, we would write, “graphics and text that instruct assembly, e.g. instruction manual.” The complete list of our representations appears in Table 1.

<b>A</b>	telephone directory (names, not yellow pages)		
		subway map	<b>B</b>
<b>C</b>	map of US cities and states		
		periodic table of elements	<b>D</b>
<b>E</b>	phylogenetic tree (e.g. of an animal or plant species)		
		Roget style thesaurus or Wordnet	<b>F</b>
<b>G</b>	logical/mathematical proof with lemmas & corollaries		
		graphics and text that instruct assembly (e.g. instruction manual)	<b>H</b>
<b>I</b>	the world-wide web (both text and images)		
		architect's floor plan drawings/blueprint	<b>J</b>
<b>K</b>	histogram/bar graph (e.g. stock market prices by month)		
		Lib.of Congress cataloging system	<b>L</b>
<b>M</b>	stores inside a 3-story mall or office locations within Media Lab		
		zip codes	<b>N</b>
<b>O</b>	a TV guide (tables showing each channel's schedule each hour each day)		
		a processing flow diagram (e.g. for a chemical plant)	<b>P</b>
<b>Q</b>	Venn diagrams		

Table 1: The 17 stimuli presented to subjects. Each stimulus describes a representation which uses a certain representational form.

Our subjects were eight MIT graduate students and one professor. They were first introduced to the method using a TM task based on a simple set of black and white circles (see Chapter 1). A week or so following this practice session we introduced the subjects to Table 1. As is standard in TM, subjects were given data sheets of stimulus pairs in the form of "\_\_\_ J \_\_\_ G \_\_\_", etc. Subjects were told that for each pair, they should first imagine using the representations specified to accomplish a task of their choosing. Then they should then imagine the two stimuli forming a sequence of some sort, and pick a stimulus from the remaining representations that would make an appropriate third element in the sequence, filling in the right-hand blank. See Table 2. We call this stimulus the right extrapolant. Subjects are asked to then reverse the process, imagining a sequence going in the other direction and choosing a stimulus for the left-hand blank (the left extrapolant). Finally, subjects choose a stimulus for the middle blank (the interpolant).

___ J ___ G ___	
___ J ___ G A	adding the left extrapolant
C J ___ G A	adding the right extrapolant
C J H G A	adding the interpolant

Table 2: The progress of a TM trial: the subject fills in first one extrapolant, then the other, then the interpolant.

The TM method also allows three other options for filling in the blanks of a trial. The first occurs when the elements of the pair are so dissimilar that the subject feels uncomfortable imagining a sequence based on them. This is the "infeasible" case. The second case is the "feasible, but no sample" case, in which the subject can easily imagine an appropriate extrapolant or interpolant, but it is not included in the stimulus set. The third case, the "dead end", occurs when either element of the original pair represents an extreme in the variability range of the chosen feature. For the extrapolant near that element, the subject would indicate "dead end."

Care was made to stress that it was the *use* of the representations which was of interest. The subjects then proceeded to form quintuples from all 136 pairs of these 17 exemplars. Because the 136 trials was too tedious for any one subject, the pairs were broken down into 9 groups of 15 trials (135). Each student then filled out a sheet with 30 pairs, two of the 15-trial groups, thus leading to two complete sets of quintuplets (a couple trial sets were adjusted to accommodate the one leftover trial). Each trial was done by two different subjects. The two sets were processed as if they were one subject who had done the experiment twice. Combining data in this way can be done legitimately as long as individual quintuplet trials are done by a single person. By combining data across subjects, we gain information about which features all subjects consider to be salient, but we typically lose the detailed

information about the features of secondary and tertiary importance to individual subjects (especially when they do not agree on these features). Also, the noise level typically increases in such data.

## Results

Figure 4 shows the z-scores of each point in the distribution of triple weights in the data (a la Chapter 3); if the data were random, these points would fall below the dotted line with 99% probability. Although our triples data contain only one triple with weight 6, the significance skyrockets because of the rarity of a weight 6 triple in a random simulation.

A trajectory map was generated by triples with weights of 3 and above. This threshold was chosen using the measure of unmatched triples and the Kolmogorov-Smirnov statistic as described in Chapter 3. The data resulted in the trajectory map shown in Figure 2 (using the algorithm described in Chapter 3). This map satisfies only 46% of the triples; we suggest that this lower percentage of satisfaction (lower than any data set based on one subject's data) arises from our data pooling method (see above). Nevertheless, the data do significantly differ from a simulated random subject (Figure 4).

Because no data chains pass through the *phonebook*, *phylogenetic tree*, *world-wide web*, or *mall* nodes (highlighted with concentric ovals), the graph can be divided into five overlapping clusters, or trajectories. Each cluster has links with a different hatching pattern. The nodes in this figure have been arranged in this figure to highlight these clusters. The links between *subway* and *phylogenetic tree*, and *phylogenetic tree* and *LC catalog* have been split in order to illustrate the separate but overlapping trajectories that begin at *world-wide web*. Figure 3 shows the same trajectory map with the trajectories separated. Several stimuli were left "floating"; they were not attached to the main body of the graph by any links above the noise level. We will explore the fate of these nodes below.

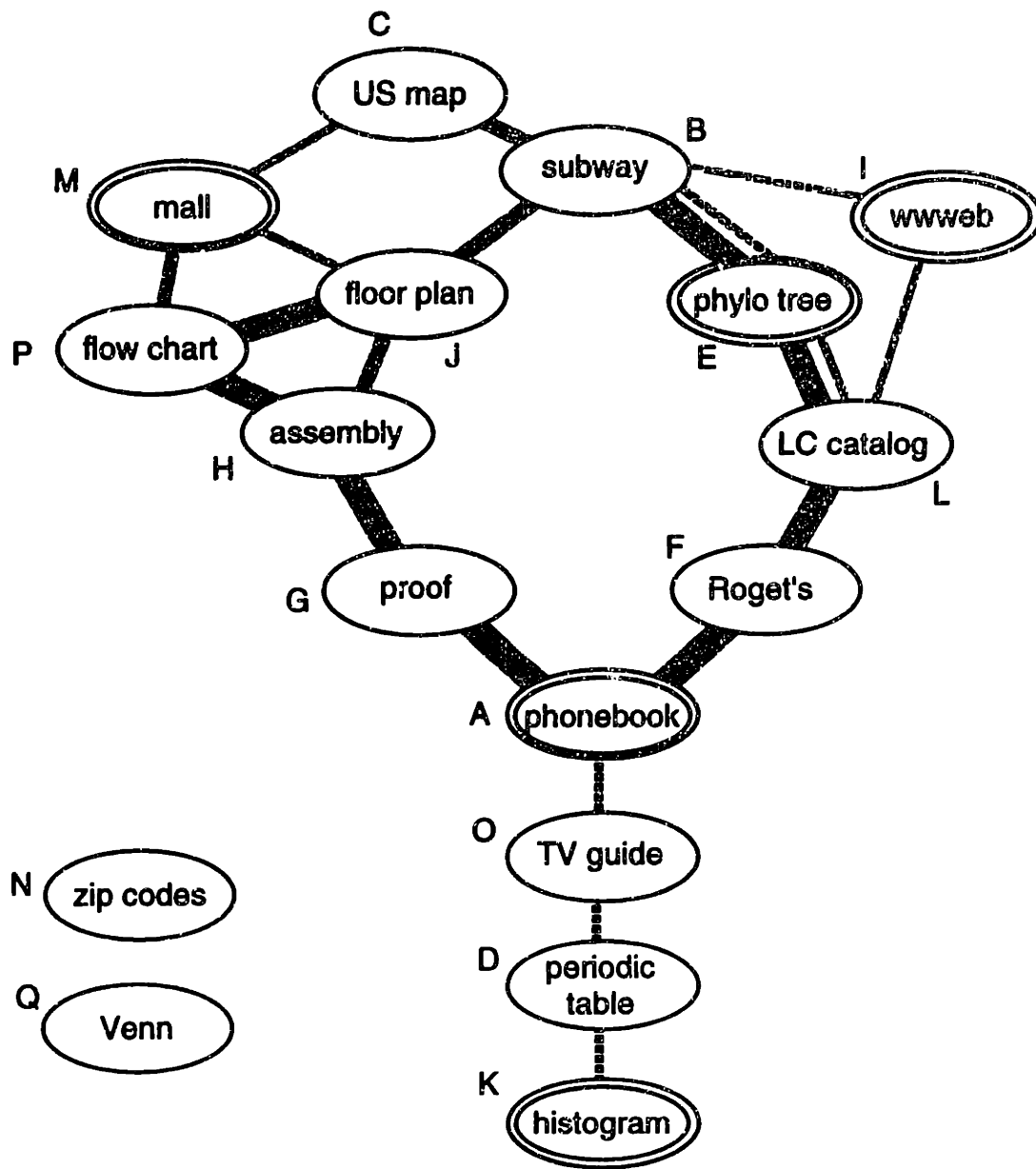


Figure 2: The trajectory map of representations. Stimuli with double ovals are "dead ends," i.e. no data paths pass through them. Three distinct trajectories emanate from *phonebook*, as denoted by the link hatching. The upper right trajectory contains static hierarchical representations, the upper left trajectory contains representations of a process or flow, and the lower trajectory contains table-like representations. In all three trajectories, complexity/dimensionality increases along the trajectory. There are also two minor trajectories beginning at *world-wide web*.

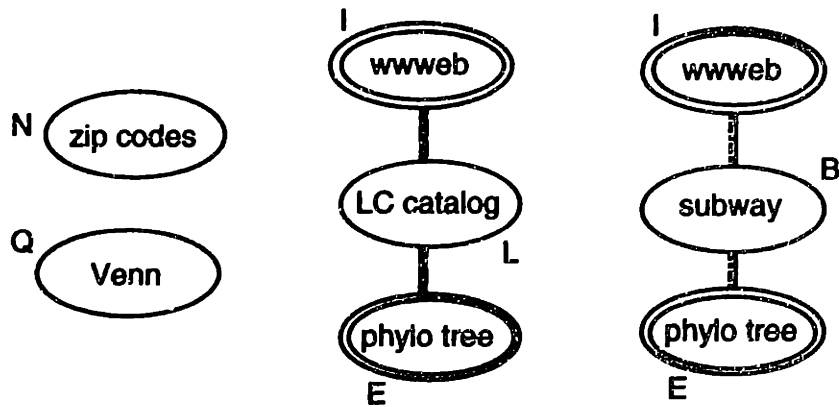
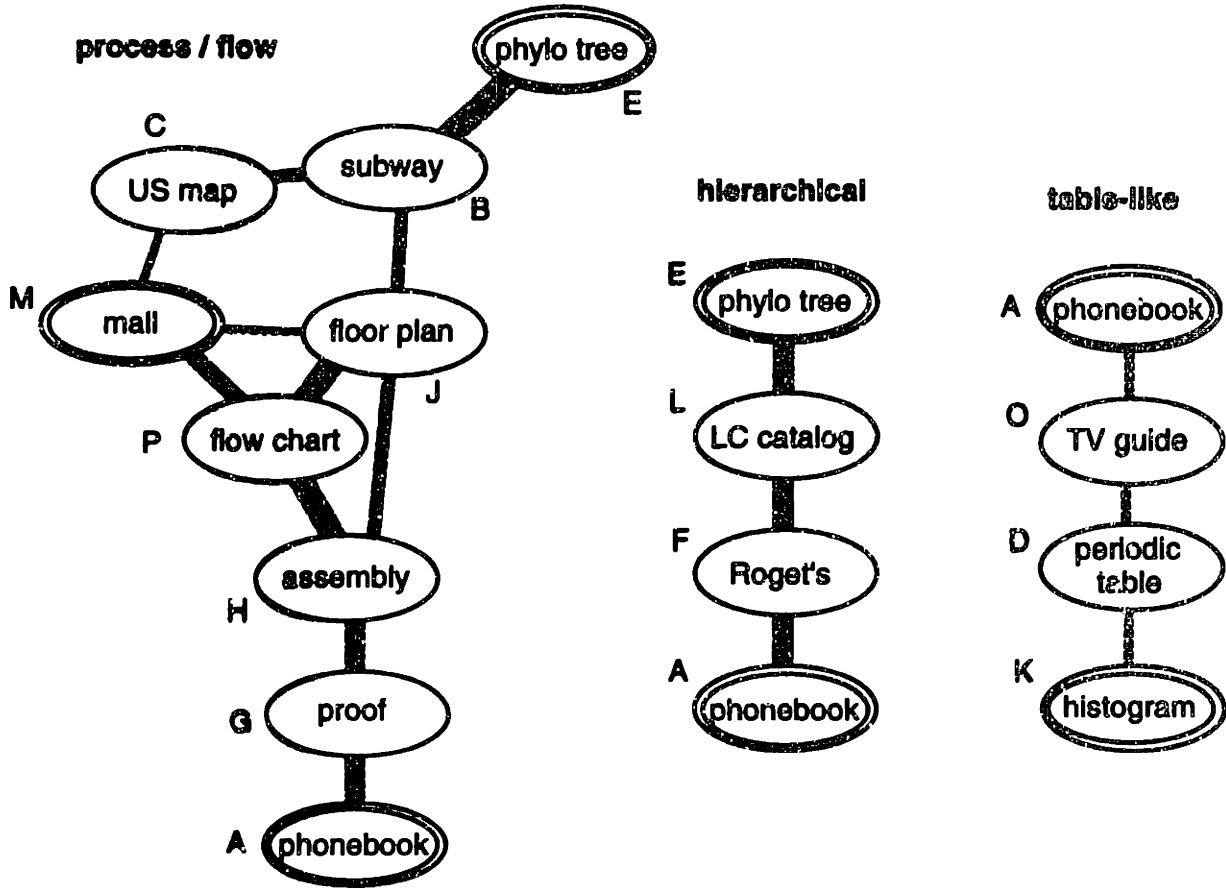


Figure 3: The trajectory map separated into separate trajectories. The three major trajectories are shown at the top.

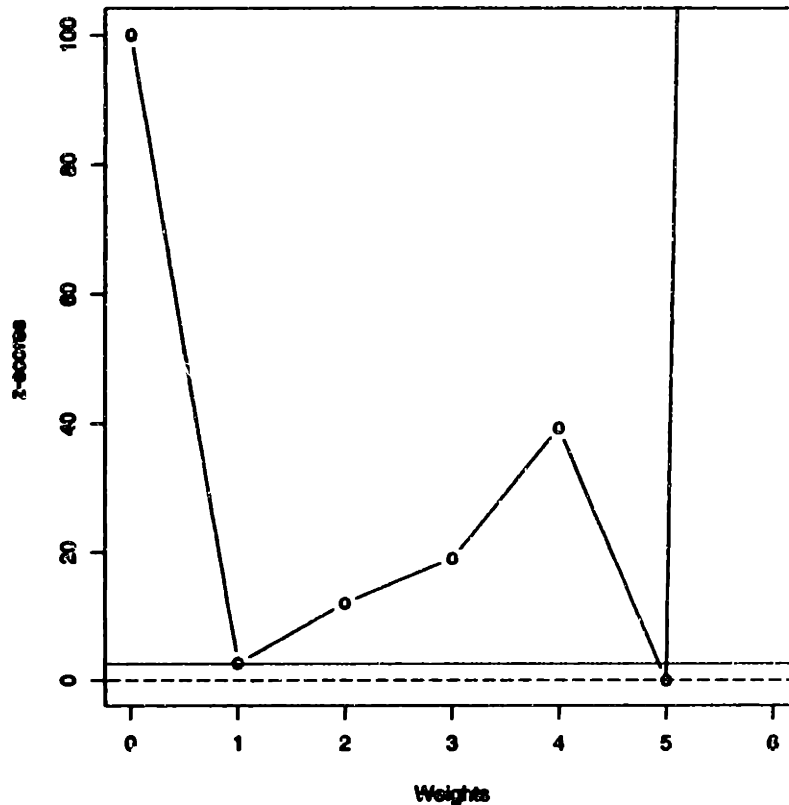


Figure 4: Z-scores of distribution of triples in the subject data show that the data differ strongly from a random subject. Random data would fall below the dotted 99% confidence interval.

The main topological features of the trajectory map are the three branching clusters emanating from *phonebook* (See Figure 2). Because of the nature of the stimuli within each of the clusters, we hypothesize that the larger cluster at the upper left (with *assembly*) contains representations of a process or flow, the smaller upper right cluster (with *Roget's thesaurus*) contains static hierarchical representations, and the lower "stem" cluster (with *TV guide*) contains tabular figures. There are also two smaller trajectories involving the *world-wide-web* and the *phylogenetic tree*. Because each trajectory is based on a single triple, we separate them from the three more salient trajectories. Nevertheless, the two small trajectories offer the interpretation that the *world-wide web* can be considered to be more path-like (adjacent to *subway map*), or more hierarchical through cross-references (adjacent to *LC catalog*).

Another important aspect of the graph is the ordering within the larger clusters. If one considers *phonebook* to be at the "root" of each cluster, then we see that the nodes within each cluster increase in complexity or dimensionality as one moves along the cluster away from the root. A *phonebook* is a one-dimensional listing of names, addresses, and phone numbers, which can be considered slightly



hierarchical if the addresses and phone numbers are subordinate to the names. In the hierarchical trajectory we move then to *Roget's thesaurus*, a list of words that is cross-referenced, to *Library of Congress catalog* and *phylogenetic tree*, which are both fully branching. Similarly, in the trajectory of process/flow representations, a *mathematical proof* and *assembly instructions* are quasi-linear sequences of transformations. The *flow chart*, *floor plan*, *subway map*, and *US map* are all two-dimensional depictions, with varying degrees of network structure. A *US map*, for example, could be a map focusing on state and county boundaries, or it could focus on the interstate highways and rail lines. Finally, a representation of a 3-story *mall* is likely either a network or a 3-D configuration of some sort.

Before discussing our basic result further, we would like to investigate the extra "floating" stimuli. There are two basic reasons why stimuli can be detached from the main trajectory map: either the subjects marked most pairs involving these stimuli as "infeasible" (they couldn't think of how to form sequences using these stimuli), or subjects used these stimuli as "good fits" in many different contexts (more than two for this data set), resulting in no highly weighted links for any particular contexts. In this particular case, no stimulus has an unusually high number of infeasible judgments. Thus, we look to the data for different contexts of linking.

The "adjacencies" in the subject data are useful for exploring this issue. By counting the number of times that each stimulus appears adjacent to each of the other stimuli in the subject quintuplets, one can build an adjacencies matrix that resembles the proximities matrix used in similarity-based techniques. (As described in Chapter 2, TM-based adjacencies tend to have a rank correlation of approximately  $0.5 \pm 0.2$  with similarities gathered for the same stimuli in separate experiments.) Figure 5 contains the adjacencies matrix for the representation data.

	phonebook	subway	US map	periodic	phylo tree	Roget's	proof	assembly	wwweb	floor plan	histogram	LC catalog	mall	zip codes	TV guide	flow chart	Venn
phonebook		2	3	10	9	11	10	4	1	1	7	13	1	22	13	0	1
subway			25	4	10	9	1	1	5	7	5	1	5	0	1	11	4
US map				11	0	2	2	1	4	11	3	1	14	10	3	1	10
periodic					5	7	1	2	1	8	7	5	3	4	13	3	8
phylo tree						13	6	6	9	0	4	12	1	4	4	6	5
Roget's							2	5	12	2	3	16	0	3	9	2	2
proof								12	2	2	5	4	0	4	2	6	3
assembly									4	24	5	0	4	0	5	20	1
wwweb										5	0	4	12	0	5	5	4
floor plan											1	0	14	1	4	9	1
histogram												2	1	3	10	1	7
LC catalog													0	12	4	4	1
mall														0	4	7	6
zip codes															6	1	2
TV guide																2	6
flow chart																	2
Venn																	

Figure 5: The adjacencies for the representational forms data. The adjacencies measure for two stimuli is the frequency of those stimuli appearing next to each other in the quintuplet data.

The adjacencies show that *zip codes* was indeed used in several contexts. The *zip codes* stimulus seems to have been considered to be a simple list of numbers by some (22 times neighboring *phonebook*), while others considered it to be a map-like (10 times neighboring *US map*), or a hierarchy (12 times neighboring *LC catalog*). The *Venn diagram* was found relatively often near *US map* and *periodic table*. In both cases we suspect that the cost of linking these three nodes in the final trajectory map was too high, since they would have to be attached to nodes that are otherwise quite distant.

To explore this issue further, we perform hierarchical clustering on the adjacencies, as if they were a similarities matrix. Figure 6 shows a tree based on hierarchical clustering of the TM adjacencies matrix using average linking (Duda & Hart, 1973). The floating stimuli are shaded gray. The tree supports our trajectory map; the three major clusters of the trajectory map are also clustered in the tree.

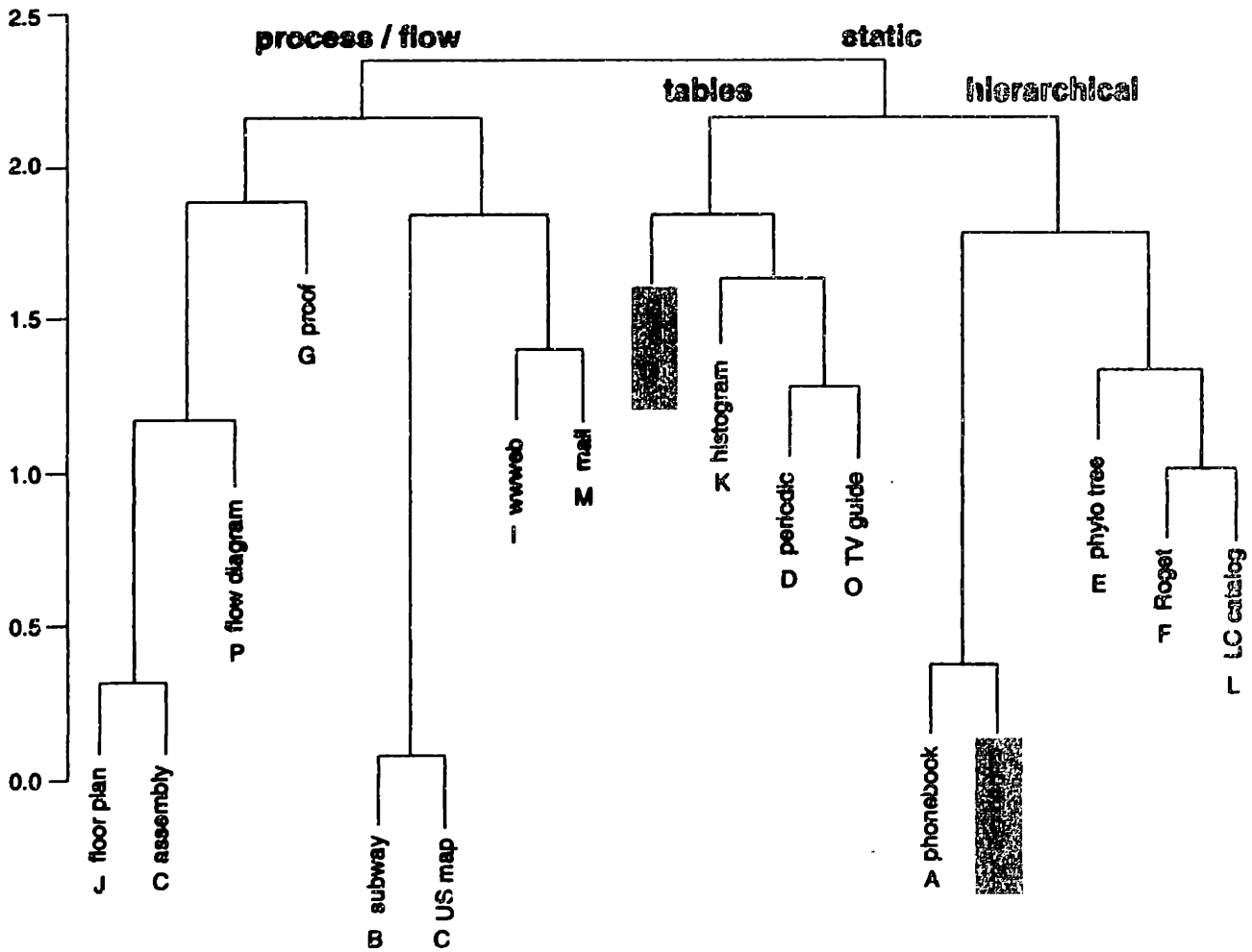


Figure 6: Hierarchical clustering tree based on adjacencies of TM data. The four "table-like" stimuli among the floating nodes of Figure 2 can be seen clustered together.

A multidimensional scaling plot (Shepard, 1962; Kruskal, 1964) of the adjacencies data (Figure 7) offers a nice framework for combining the information from the above two figures. Here we can see both the trajectories (links) and the three major clusters from the tree (gray shaded) again. We do not draw the smaller trajectories that include the *world-wide web* for the purpose of clarity. The trajectory of hierarchical representations lies "behind" the *phonebook to math proof* link according to the multidimensional scaling (MDS) plot in 3-D, which shows each of the three clusters occupying a part of the surface of a rough sphere. We hesitate to draw many conclusions about the data from the MDS plot itself because of its high stress (0.52 in 2-D, 0.41 in 3-D), but even at this high degree of instability, one can see that the x-axis can be described as a linguistic-pictorial dimension.

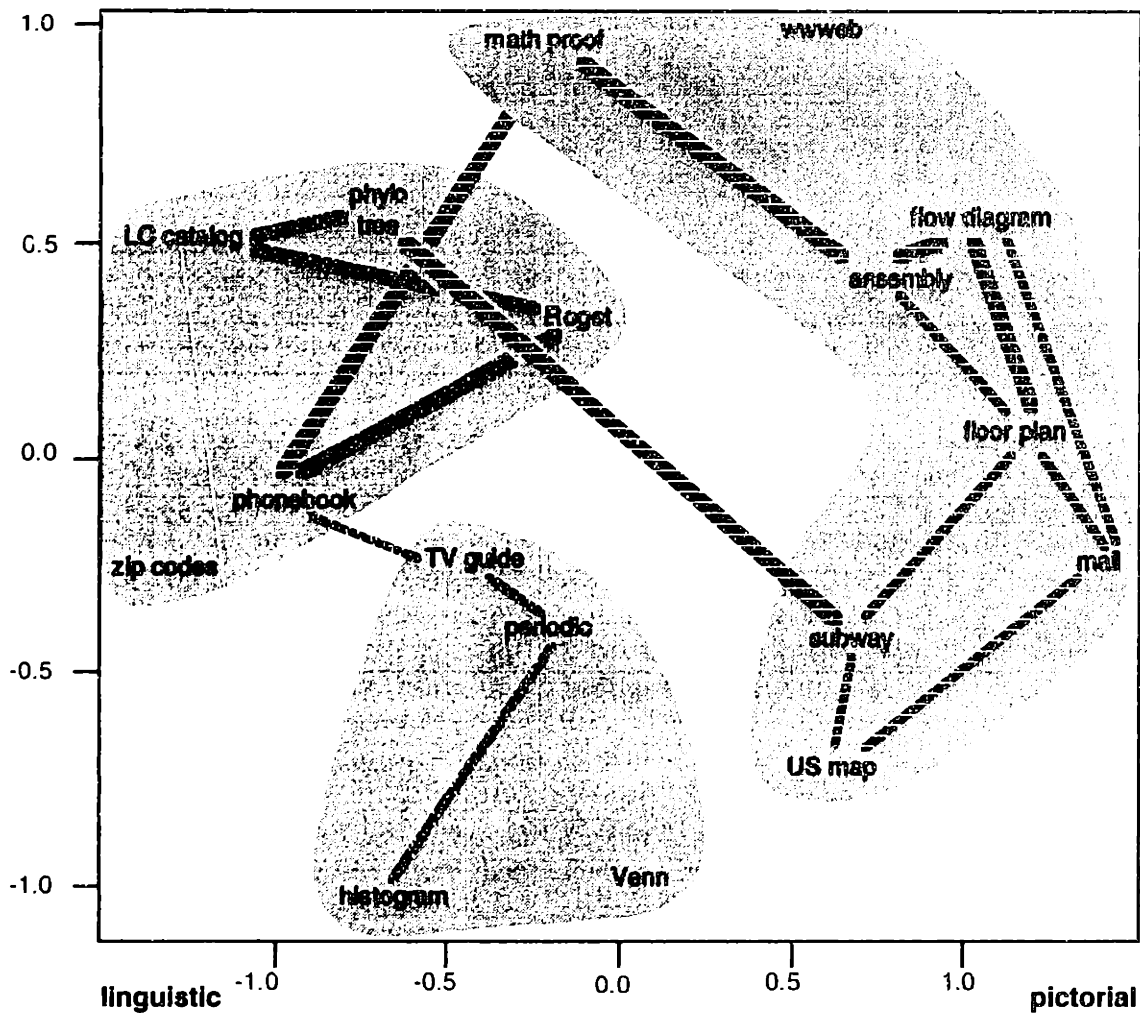


Figure 7: An MDS plot of the adjacencies data offers a framework for combining all three representational forms, the trajectory map, the clusters, and the MDS metric space. Forms can be seen to vary from linguistic from pictorial along the x-axis.

## Discussion

These results lead us to two salient conclusions. The first is that despite the importance of work that has been done exploring the cognitive differences in processing pictorial vs. linguistic representations (Cox & Brna, 1993) or diagrammatic vs. sentential representational forms (Larkin & Simon, 1987), subjects did not find these characterizations most valuable for grouping our 17 representations. (The process/flow trajectory of representations contains at least one for each of these characterizations.) Related to this idea is the importance of "mixed" representations, i.e. those containing both graphics and text. The process/flow trajectory contains the complete range of pictorial-linguistic axis; only the most outer forms are close to pure; most are mixed. Likewise, the linguistic-pictorial x-dimension of the MDS plot does not reveal that representations are grouped on one side or the other. Representational forms could also be mixed (in terms of diagrammatic expressions and sentential expressions), but these seem rarer. An

example is the proof of the Pythagorean theorem described by Tufte (1990) that includes both a verbal argument that includes graphics and a main graphic as well.

The second conclusion we offer is that when classifying representations for the purposes of use, rather than for graphic or computational purposes, subjects use features from all three levels of analysis described earlier: computational structure, external or graphic structure, and functionality. The trajectory of process/flow forms is a functional category. The hierarchical trajectory makes up a computational category, and the tables are a graphic, external depiction category.

As an illustration of the differences between the categorizing done by our subjects and a traditional categorization, we introduce a classification matrix based on Twyman (1979). His original matrix, designed to classify "graphic language," proposed 28 different categories laid out in a 4 x 7 matrix. The distinguishing dimensions were linear to non-linear (seven categories) and verbal to pictorial (four categories). The verbal to pictorial dimension included mixtures of both, plus sketches or diagrams. The linear to non-linear dimension included lists, branching structures (such as both textual and pictorial trees), tables, and multiple scale representations, such as a newspaper where there are large headlines and separate articles in small text, or pictures with varying levels of detail. Examples of the 28 cells of the matrix were provided, although some were quite uncommon or "strained". Figure 8 shows a slightly abridged version of Twyman's scheme. (Here the schematic and picture distinctions are merged, as well as two non-linear distinctions and the pure and interrupted linear distinctions.) Our stimuli have been placed in the cells as appropriately as possible, and the cells have been hatched by their exemplars' cluster in the trajectory map.

	linear	list	branching	matrix/ table	non-linear
verbal/ numerical	math proof G	phone book A	phylo/catalog E, L	TV guide C	(front page news)
verbal & pictorial	(comic strip)	assembly H	flow diagram P	periodic/floor D, J	www I
pictorial & schematic	(music notation)	(airline emerg. brochure)	(org. chart w/o text)	histogram/Venn K, Q	maps, mall B, C, M


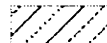

 = process/flow     
  = hierarchy     
  = tables

Figure 8: A classification matrix for graphic representations based on Twyman. When the 17 representations are placed within it, along with hatching according to their cluster, one can see by the patterns of hatching that subjects clustered based on three different types of characteristic: computational, functional, and external (or pictorial).

The patterns of hatching illustrate well the differences in the level of categorization of the three clusters in our data. The trajectory of process/flow stimuli, because they share mainly function in common, instead of any external attribute of Twyman's rows or columns, snakes through the table from corner to corner, not fitting in any one column or row. This diagonal path shows that the trajectory includes graphic and linguistic representations, as well as mixed ones (Twyman's rows). If Twyman's columns can be seen roughly to increase in complexity and dimensionality, the process/flow trajectory can be seen to span the range of complexity here just as in the trajectory map. Our hierarchy trajectory can be appropriately seen to occupy mainly the "branching" column of Twyman's matrix, a category which was intended as a graphic characteristic, but which could also describe the computational nature of a hierarchy. Lastly, our tables can be seen to occupy the matrix/table column, a very good match since our term "tables" is a pictorial description.

### **Conclusion**

Our original intent was to examine which type of characteristics subjects would use to organize representations for efficient use: computational, functional, or external. Our main result of dividing the representations into process/flow representations, hierarchical representations and table-like representations suggests that when we organize for use, we use all three of our suggested characteristics of representations to do so. This result offers an important contrast to classification work that focuses more on one characteristic, assuming that the researchers are interested in more features than just those that we use to organize the representations graphically.

Another approach to explore how subjects organize representations for use is examine how they switch from one representation to another during problem solving. Cox, Stenning, and Oberlander (1995) have some preliminary data on this question. They analyze "work scratchings" of subjects solving GRE analytical problems. Their main result is finding individual differences among students: one type tended to use a linear approach, whereas the others proceeded piecemeal with subcomponents of the problem resolved first, and then these components were integrated (i.e. a divide-and-conquer strategy). They also studied the way students switched from one representation to another when offered a variety of computerized representations. This work has been limited to only a few GRE problems so far, however, and we must wait to make any general conclusions.

They, like Cox & Brna (1993) however, categorize students' work scratchings as either pictorial or linguistic. It would be worthwhile to carry out further such experiments on representation switching and analyze whether subjects share this mental categorization of the representations. Our results suggest that even if they used the labels "pictorial" and "linguistic" as extremes in their space of representations, they would include a wide variety of mixed representations in between them.

## References

- Bertin, J. (1967) *Sémiologie Graphiques, 2d ed.* Gauthier-Vallars, Paris. (English translation: Berg, W.J. *Semiology of Graphics.* Madison: Univ. of Wisconsin Press, 1983).
- Cox, R. & Brna, P. (1993) Analytical reasoning with external representations: The relationship between prior knowledge and performance. In R. Cox, Petre, M., Brna, P. and Lee, J. (Eds.), *Proceeding of workshop on pictorial representations, reasoning and communication* held at World Conference on Artificial Intelligence and Education (AI-ED93), August, Edinburgh, 33-36.
- Cox, R., Stenning, K. & Oberlander, J. (1995) The effect of pictorial and sentential logic teaching on spontaneous external representation. *Cognitive Studies: Bulletin of the Japanese Cognitive Science Society*, 2(4), 56-75.
- Duda, R.O. & Hart, P.E. (1973) *Pattern classification and scene analysis.* New York: John Wiley & Sons.
- Kruskal, J. B. (1964) Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29, 28-42.
- Larkin J.H. & Simon, H.A. (1987) Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, 65-99.
- Lohse, G.L, Biolsi, K., Walker, N., & Rueter, H.H. (1994) A classification of visual representations. *Communications of the ACM*, 37(12), 36-49.
- Moray, N. (1990) A lattice theory approach to the structure of mental models. *Philosophical Transactions of the Royal Society of London B*, 327, 577-583.
- Norman, D. (1993) *Turn signals are the facial expressions of automobiles.* Reading, MA: Addison-Wesley Pub. Co.
- Richards, W. & Koenderink, J. J. (1995) Trajectory mapping: A new non-metric scaling technique. *Perception*, 24, 1315-1331.
- Shepard, R. N. (1962) The analysis of proximities: Multidimensional scaling with an unknown distance function. I. & II. *Psychometrika*, 27, 125-140, 219-246.
- Tufte, E.R. (1990) *Envisioning Information.* Cheshire, Connecticut: Graphics Press.
- Twyman, M. (1979) A schema for the study of graphic language (Tutorial paper). In Kolers, P.A., Wrolstad, M.E., & Bouma, H. (Eds.), *Processing of visible language, Vol. 1.* New York: Plenum Press.
- Winston, P.H. (1980) Learning and reasoning by analogy. *Communications of the ACM*, 23(12), 689-703.





## Chapter 6

### Conclusion & Summary

#### Introduction

Whenever communication takes place, each participant builds a model of the other participant's "mindset" in order to choose the most meaningful form of knowledge representation. Put in an everyday context, this mutual modeling means that one might use less complicated words when talking about life with a 3rd-grade cousin, because one's model maintains that he wouldn't otherwise understand. It means that one doesn't tell the clerk at 7-11 about a friend's favorite CDs, because one's model suggests that she is probably not interested.

To perform this type of modeling artfully is to communicate effectively, but unfortunately our modeling abilities tend to vary widely across individuals and even across content domains within the same individual. Our poor performance in modeling the "mindset" of others manifests itself in many aspects of our lives, from the best-selling pop-psychology book *You Just Don't Understand* (Tannen, 1990) to students who fall behind in the classroom because the teacher doesn't provide examples expressed in structures that they understand.

This thesis addresses an aspect of this topic by offering an algorithm for exploring the different representational forms that people use to structure their knowledge. The same individual typically uses different forms for different types of knowledge, and different individuals often use different forms for the same knowledge. For example, if a friend were to invite you to a party at his house, you might request information about how to find it. The friend could offer a variety of representations for navigating to his house. He could give you a street map or a hand-sketched personalized map. He could give you directions based on landmarks, or directions based on North, South, East, and West.

Each of the forms of navigational aid are a different representational form of the same information. If the friend were clever, he would offer you the form that you are most accustomed to using; part of his model of you might be, "This person likes to navigate using landmarks." This cleverness is also a goal that we can aspire to in the realm of technology; if the technology designer is clever, she will offer an interface that anticipates the various mental representations held by the user. The same is true in fields such as digital libraries, where it is important to index data using the appropriate representations for people, so that they can easily find the information that they seek.

### **More Specifics: Spaces, Clusters, and Networks**

In this field of perceptual and conceptual scaling, researchers have focused on two particular representations for knowledge: *metric spaces* and *hierarchical clusters*. An example of a metric space is the 3-dimensional space of all colors of light, where the axes are red, green, and blue. Each color can be defined by specifying a certain amount of red, green, and blue. More generally, in a metric space, items have coordinates in an n-dimensional space, where each dimension is some feature of the objects. The distances between the objects can be seen as inversely proportional to their similarity to each other.

An example of a hierarchical clustering representation is the Macintosh file system, in which files sit within folders, which sit within other superordinate folders, which sit within a disk drive, which sits on the Desktop. In hierarchical clusters, each object belongs to a single cluster (overlapping clusters are not allowed), and each cluster belongs to a single superordinate cluster, except the root cluster, which stands alone and includes that entire set of objects. Hierarchical clusters are often represented graphically by trees.

A given set of stimuli can be arranged in either of these representations, and a researcher can choose the representation according to the prior assumptions about the structure of the data. If these two representations completely spanned the space of all possible representations, then this forced choice would pose only the problem of deciding on the representation. The two representations leave a significant gap in the representation space, however, which is the network or connected graph.

Metric spaces and hierarchical clustering also have the disadvantage that they are traditionally constructed from similarity data. Using similarity creates a variety of difficulties (Shepard, 1974), such as their assumption that stimuli are symmetrically similar, i.e. A is just as similar to B as B is to A. Secondly, similarity judgments are often ambiguous; it is unclear which features a subject uses to choose the similarity value.

### **What's New: The Contributions of Trajectory Mapping**

To help alleviate the difficulties associated with similarities and to offer an approach that uses the traditionally ignored network representation, this thesis shows the advantages of using Trajectory Mapping (TM) as a complementary or supplementary technique. By using the connected graph or network, TM allows different features of a stimulus each to play a role; contexts set by other stimuli can intersect. Thus, TM can often tease apart the features that otherwise would be bundled together in subjects' judgments of the most generic feature, similarity. Directed graphs also follow as a natural extension, removing the assumption of reflexive symmetry in sequencing judgments.

TM also offers some of the advantages of both older techniques. Trajectories can be seen as clusters of stimuli, revealing features much like hierarchical

clustering. The stimuli within the trajectory clusters are ordered, suggesting features much like MDS. Lastly, the adjacencies in the TM data serve as an approximation of the same sort of proximities matrix that the older methods require (Chapter 2).

Because the original TM approach (Richards & Koenderink, 1995) did not supply an objective algorithm for constructing trajectory maps, we have offered such an algorithm (Chapter 3). Based on the simulated annealing, the algorithm uses triples derived from subject data as constraints that can be used to find an optimal connected graph. We have chosen parameters of the cost function so that the algorithm models the manual heuristics followed by Richards & Koenderink. The computer code for the algorithm and most processing steps can be found in Appendix C. It is also available from the author and soon to be online.

After comparing trajectory maps constructed manually with trajectory maps constructed by the algorithm, we find them to be similar: tests in five domains of data revealed a positive correlation between 0.42 and 0.80 for all sets (Chapter 4). With simpler data sets a human being can sometimes do better than the algorithm through a better sense of the conceptual structure of the data (e.g. kinship terms: manual solution matched 87% of triples, algorithm only 70%). For more complex data sets, however, the algorithm can often provide a clearer solution (e.g. London subway loop: manual solution matched 95%; algorithm matched 99%).

Inspired by the issue of the degree to which metric spaces, hierarchical clusters, and trajectory maps span the space of representational forms, we collected subjects' judgments about 17 representations (Chapter 5). Our goal was to model the subjects' mental organization of these representations for the purposes of their use. Taking advantage of all three of the scaling methods discussed above, we concluded that subjects classified the 17 representations into three main categories, each according to a different characteristic of representations. The process/flow trajectory is a functional, content-based category; the hierarchical trajectory is a computational category; and the table-like trajectory is a depictive category. The data also emphasized the importance of "mixed" representations, i.e. representations that consist of both pictorial and textual elements.

## **Summary**

The most salient contributions of this work are: 1) a thorough analysis of Trajectory Mapping as a scaling technique, 2) an algorithm which makes Trajectory Mapping a more objective technique, 3) both a theoretical and data-driven comparison of various scaling techniques, and 4) experimental results that both suggest the importance of representations with a mixture of textual and pictorial elements and characterize representations at multiple levels of abstraction.

## References

- Richards, W. & Koenderink, J. J. (1995) Trajectory mapping: A new non-metric scaling technique. *Perception*, 24, 1315-1331.
- Shepard, R.N. (1974) Representation of structure in similarity data: Problems and prospects. *Psychometrika*, 39, 373-421.
- Tannen, D. (1990) *You just don't understand: Women and men in conversation*. New York: Morrow.

## Appendix A

### Deeper Analysis of the TM Algorithm: Work in Progress

In this appendix, we discuss various issues surrounding TM and the TM algorithm that we have not yet resolved or which are still work in progress.

#### Part I: Diagnostics of TM Subject Data

There are four useful parameters to consider when analyzing TM input data. These parameters are calculated from the triples that are derived from the TM quintuplets.

##### Top Weight

The first measure is the top weight, i.e. the highest weight of the set of triples, normalized by the maximum possible weight. It turns out that for  $N$  stimuli, the maximum frequency a triple can have is  $N$ . Thus the top weight is a number  $(0,1]$  that gives a measure of how strong a pattern is present in the triple data. If there is a strong pattern, the top weight will be high, and triples will be distributed in exponentially growing numbers (see measure of randomness, Chapter 3). If the data set has no real pattern, random noise will generate some weights of 2 and sometimes 3, but top weight will be low. In our experience thus far, a top weight greater than 0.4 indicates a strong pattern.

It is unclear, however, what it would mean if the distribution of weights is simply shifted higher. The two sets of triples below, for example, result in the same trajectory maps with almost identical weights on the links.

w 3	w 5
1 2 3	1 2 3
2 4 5	2 4 5
w 1	w 3
2 3 5	2 3 4

##### Degree of Conflict

The second measure is the degree of conflict in the triples. Given a triple such as (1 3 4), then we say that (1 4 3) and (3 1 4) conflict with it. Triple (4 3 1) does not conflict; it's simply a reversal. Triples conflict when it would be impossible to have three nodes that are ordered according to both triples simultaneously. We are unsure whether to try to normalize by the maximum possible number of conflicts, which changes complicatedly with  $N$ .

### **Variance of Triples**

The next measure is the variance of the distribution of triple weights. If the variance is high, we can assume that there is more noise in the data, that is, that the subject has a certain structure in mind (the “signal”) and that one of two cases is true. Either the subject is uncertain about the structure and allows the non-important stimuli to distract him, or he sees a weak structure in some of the stimuli in addition to the most salient structure. Both of these cases would lead to a set of triples with low weights alongside a set of triples with higher weights. A set of simulations is required to confirm this phenomenon.

## **Part II: Noise and TM**

In an effort to examine how robust the TM algorithm is with respect to noise, we have analyzed the trajectory maps that result from fitting data that is generated with different amounts of noise from a pre-defined trajectory map. We tested under two different levels of homoschedastic noise, and one type of heteroschedastic noise (that varied across the generative map from low to high). The main result of the noise is various shifts in the ordering of stimuli within the graph; the overall shape of the graph remains the same, however.

Another type of data interference occurs if a subject violates one of the two assumptions of TM. As noted above, the first assumption is that the subject should use the same features when comparing stimuli within a given trial. Secondly, the subject should choose extrapolants and an interpolant that are appropriately spaced.

To investigate the problems that arise when a subject does not use the same feature within a given trial, we note the quintuplet below. It is conceivable that in a fruit and vegetable scaling experiment, a subject could have chosen this quintuplet with the following features in mind: going from apple to orange to banana, “color”; going from orange to apple to bell pepper, “lobeness in shape”; and going from orange to pear to apple, smoothness of peel. If the subject consistently chose stimuli using a variety of features like this, then it is unlikely that any heavily weighted triples would arise.

bell pepper      apple      pear      orange      banana

Violating the spacing assumption creates a similar result as the noise described above; depending on the degree of violation. With violations on only some trials, the model will likely have the correct overall shape but have some nodes interchanged. With consistent violations, the model will likely be quite distorted.

### Combining Subject Data

We know that combining subject data can be a useful method of highlighting the features common to both subjects, although it adds noise to the less salient features, but we have not done a complete analysis of data pooling.

### Part III: The Parameter Weight Space

The graphs of the varying graphs in the cost function parameter space (Chapter 3) suggest the three parameters that we varied might be able to be represented by a single parameter, rendering the process of estimating the optimal weight space less complicated. The algorithm would also likely require less processing time, since it would iterate only over one dimension. This potential should be explored.

Since our algorithm's task is a combinatorial optimization problem, the parameter space is not uniform; there are a variety of phase transitions. As one moves through the phase plateaus, the optimal graph does not change. This parameter space should be fleshed out thoroughly.

### Part IV: Ambiguities in the graph representation

Unfortunately, the original choice of a representation for trajectory maps includes some ambiguities. Rather than present an alternate representation in this paper, we will simply note the ambiguities and the ways of getting around them.

#### Branching vs. Intersection

Consider the graph below in Figure 1. Node 3 could be the hub of three different branching trajectories that all meet at 3 and find a dead end. Or, there could be two trajectories, the longer of which is 6 - 5 - 4 - 3 - 7 - 8.

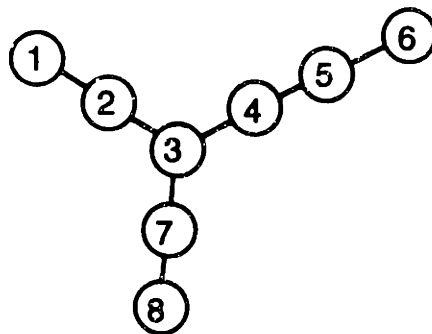


Figure 1

### Cycle vs. Parallel Paths

Similarly, ambiguity can arise when confronted with graphs such as the one below in Figure 2. Should one consider it to be a cycle with a spur, in which one trajectory is 1 - 2 - 3, and then another is the cycle 3 - 8 - 7 - 6 - 5 - 4 - 3? Or is it two parallel and somewhat overlapping trajectories, the first being 1 - 2 - 3 - 4 - 5 - 6 and the other being 1 - 2 - 3 - 8 - 7 - 6?

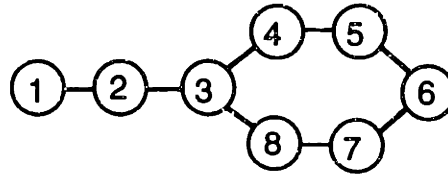


Figure 2

### Resolving the Ambiguities

The thresholding process can often indicate the roles of different branches as they break apart. There are several more subtle methods of identifying these roles, however. One way is to tag the nodes which the subject considered to be dead ends. In all following graphs in this paper, nodes which are seen as dead ends in any quintuplet will be drawn as a diamond instead of an oval or circle. This standard usually solves the cycle vs. parallel path ambiguity, in that if node 6 is a dead end and node 3 is not, then one would lean towards the parallel path interpretation in Figure 2. The dead ends do not usually solve the branching vs. intersection ambiguity, however, because even if node 3 is a dead end in Figure 1, it might play that role only in the 8 - 7 - 3 trajectory, while being just a normal node in the 1 - 2 - 3 - 4 - 5 - 6 trajectory.

To solve this latter problem, one can confer with the original list of triples. If there are no triples which span node 3, then it is likely a dead end for all 3 branches. If you do find heavily weighted triples like 2 - 3 - 7 or 1 - 3 - 8, then one can hypothesize that those trajectories join to form a longer path.

We believe that future uses of TM will include directed links, which would assist in resolving the ambiguity, especially that of the cycles.



## Part V: Future Work

There are many possibilities for honing this approach. Some are more subtle suggestions for future work, while others would change the TM approach significantly. First we describe the more subtle points.

### Counting Triples

An obvious issue arises from how we count our triples from the quintuplets. To review, when we see quintuplet (A B C D E), we consider the triples to be (A B C), (B C D), (C D E). This makes sense if the quintuplet is a continuous chain of stimuli, but since subjects often do not stop to check the full fit of each quintuplet, using the triples that the subject actually considered, triples (A B D), (B D E), and (B C D), might be more appropriate. We have found in brief pilot experiments that this different counting method does not matter enormously, but we must do additional tests to confirm this hypothesis.

### Spacing Assumptions

We currently ask subjects to choose stimuli for the quintuplets such that they are as equally spaced as possible. Because we explain the goal of having an equally-spaced quintuplet, we assume that all elements of the quintuplets are equally spaced in the subject's mental map, like this:

A      B      C      D      E

Depending on the subject's interpretation of our instructions, however, it could be that only the triples that the subject considers are equidistant, resulting in a quintuplet such as:

A              B      C      D              E

Josh Tenenbaum (personal communication) has suggested another alternative to the equal-spacing assumption. Instead of asking subjects to choose samples which are equal spaced from one another, we should ask subjects simply to pick the sample which would represent the nearest sample to the pair. This instruction might provide less ambiguous results.

An obvious issue arises from how we count our triples from the quintuplets. To review, when we see quintuplet (A B C D E), we consider the triples to be (A B C), (B C D), (C D E). This makes sense if the quintuplet is a continuous chain of stimuli, but since subjects often do not stop to check the full fit of each quintuplet, using the triples that the subject actually considered, triples (A B D), (B D E), and (B C D), might be more appropriate. We have found in brief pilot experiments that this different counting method does not matter enormously, but we must do additional tests to confirm this hypothesis.

Josh Tenenbaum has suggested that instead of using the equal spacing assumption, that is, that subjects automatically choose extrapolants that are equally spaced from the pair as the pair elements are from each other, we

should ask subject simply to pick the sample which would represent the nearest sample to the pair. We would take this instruction into account within the simulated annealing cost function.

We have considered asking subjects for confidence judgments along with their extrapolations and interpolations so that we might use them to make better judgments of the mental distance between the stimuli. The TM process is already laborious enough, however.

We want to investigate other experimental methods for collecting the triples data. We wonder whether subjects really need to undergo all the TM ordering trials, or whether the results be similar if we instructed them, "group the stimuli as you wish, and then order each group by a salient feature." Perhaps the results would be similar for some types of data, but not others. If this were true, we could better estimate the usefulness of applying TM to a given data set before applying it. Experiments should also explore whether asking subjects only for triples instead of quintuplets would change results. This ability would be important for distinguishing feature trajectories that do not span more than three stimuli within a small stimulus set. Currently, the quintuplet format makes identifying such trajectories impossible.

It is important to try to expand TM to accommodate directed graphs. Only with directed graphs can we handle asymmetric sequencing judgments. The adjacencies from a directed trajectory map would then generate the asymmetric proximity values, and we could explore the extent to which TM models contain asymmetric similarity judgments implicitly.

The algorithm should take advantage of dead ends and infeasibles to help constrain the graphs.

## Appendix B

### Additional TM Data Sets

In this appendix we discuss three additional data sets that still require further analysis. One data set is for Boston tourist attractions (cited in Lokuge, Gilbert, & Richards, 1996). Another is a data set of historical events, used in coordination with a class of undergraduates studying the social implications of the events. The third is the data set of visual textures described in Richards & Koenderink, 1995; we suggest an alternative analysis of that data.

#### Boston Tourist Sites

This data set was chosen as an appropriate test for TM's ability to illustrate valuable information about conceptual stimuli, as opposed to perceptual stimuli. Subjects were given a written list of 15 tourist attractions in Boston. Subjects had all lived in the area long enough to be familiar with the attractions. We illustrate one subject's trajectory map in Figure 1 because of its interesting structure.

The stimuli were *Sports Museum*, *Children's Museum*, *Science Museum*, *Aquarium*, *Swan Boats* (boat tour of Public Gardens), *Newbury Street* (elegant shopping), *Quincy Market* (outdoor historic mall), *Trinity Church* (historic site), *Magic Show*, *Salem* (nearby historic town), *Harvard University*, *Museum of Fine Arts*, *Zoo*, *Fenway Park* (baseball stadium), and *Arboretum* (nature preserve).

This trajectory map satisfies 83% of the triples, a successful fit. It is interesting to note the structure; although all stimuli are in the same trajectory or cluster, there are distinct dead ends in the graph. No triple data flow through *Zoo* or *Harvard*. There are also several "corners", or paths through intersections that are not taken. They are marked with a  $\otimes$ , i.e. *Trinity Church*, *Salem*, *Harvard*; *Fine Arts Museum*, *Quincy Market*, *Swan Boats*; and *Quincy Market*, *Swan Boats*, *Aquarium*. Triples do flow across all other intersections completely.

In Figure 2 we show the same trajectory map if we duplicate the dead end stimuli at the gray bars and unfold the map. Various feature categories can be found in the map, such as "outdoor vs. indoor", "for adults vs. for children", and "active vs. passive" activity. Detailed analysis can be found in Lokuge, Gilbert, & Richards (1996).

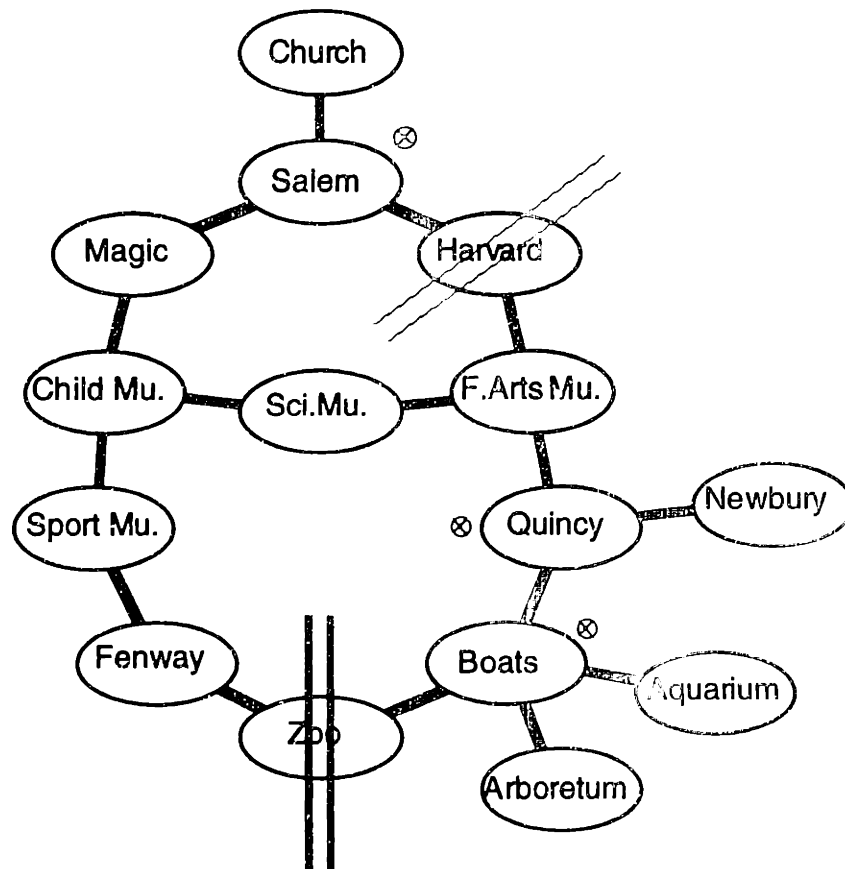


Figure 1: Trajectory map of Boston tourist attraction data. It is interesting to note that although the stimuli are all on the same trajectory, subject orderings do not flow across *Zoo* or *Harvard* (gray bars). They also do not flow through intersections on the side that they are marked  $\otimes$ , e.g. from *Fine Arts Museum* to *Quincy Market* to *Swan Boats*.

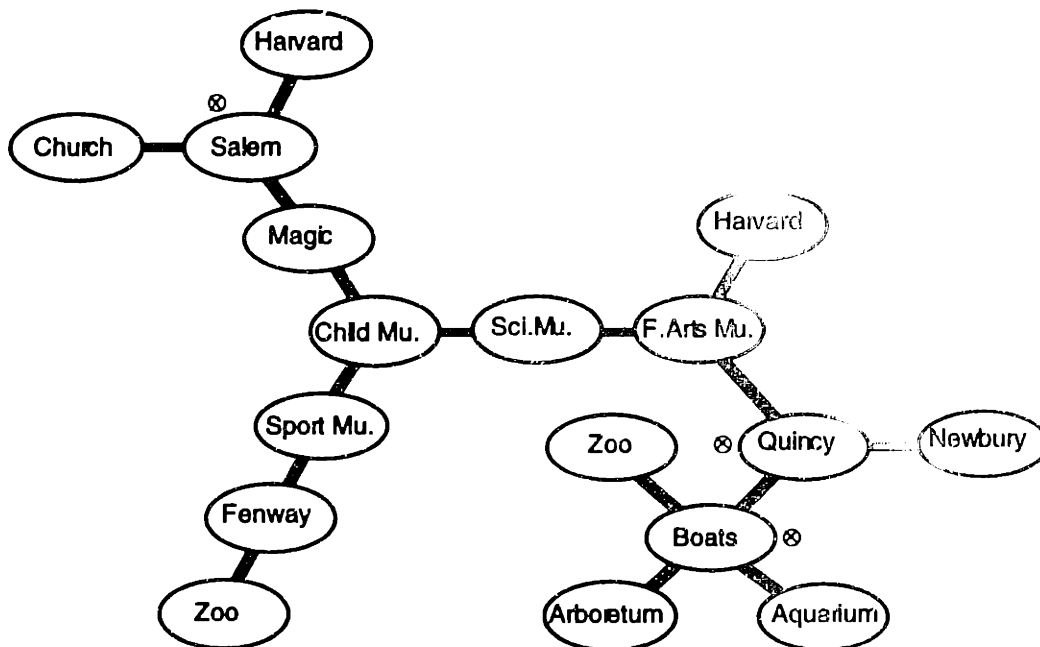


Figure 2: The same trajectory map as in Figure 1, cut at the gray lines and duplicating the dead end stimuli *Harvard* and *Zoo*.

## Historical Events

In an effort to explore whether TM might be useful at detecting changes in knowledge, we collaborated with Professor Stephan Chorover at MIT, who teaches a course that includes a segment in which students explore a large and detailed historical timeline using software written by Chorover. Part of their work with the timeline is to write an essay about the interrelationships they say between historical events.

We hypothesized that if we had students do TM historical events that seemed unrelated to the layman, then their trajectory maps would be different before and after the course, assuming the interrelationships they studied applied to these events. Using historical events as stimuli would also be a test for the ability of TM to elucidate information in a set of extremely multi-featured and abstract stimuli. The events are quotations from the timeline software.

Unfortunately, because we wanted the students to be able to perform the TM relatively quickly (20-30 minutes), we had to limit the number of trials, which limited the number of stimuli to 7. As we later discovered based on other data sets, 7 stimuli is really too few to get a clear idea of features in the resulting trajectory map, especially one of complex stimuli. We will not present the data here, since they still require further analysis, but we present the stimuli (Table 1) to document the high level of abstraction that we expect TM to handle.

- A The oriental yang/yin philosophy of nature is founded by Chinese emperor Fu Hsi. It says that health and tranquillity require perfect equilibrium.
- B More than 200,000 people of all races march for civil rights in Washington, D.C. Speaking from the steps of the Lincoln Memorial, Dr. Martin Luther King tells of his dream.
- C The National Audubon Society is founded to protest the commercial hunting of birds, and more generally, the indiscriminate slaughter of U.S. wildlife.
- D In the most destructive U.S. earthquake since San Francisco 1906, the Bay Area is shaken by a quake measuring 7.1 on the Richter scale. In all, nearly a hundred people are killed. Property losses are estimated at around \$6 billion.
- E Catherine Genovese is beaten and murdered in the streets of New York; 38 people hear her screams and see it happen but don't do anything.
- F A new copper smelting plant without emission controls is scheduled for start-up soon in Mexico, not far from the U.S. border. Despite complaints, the EPA makes no immediate plans to take any action.
- G Smallpox kills an estimated 60 million Europeans in the 18th century.

Table 1: Historical events used for TM project with Professor Stephan Chorover.



The trajectory map in Figure 3 gives an idea of how complicated the data set is. This map has an unusual number of small discrete clusters and intersections where triples travels one only one or two of the possible paths through the intersection.

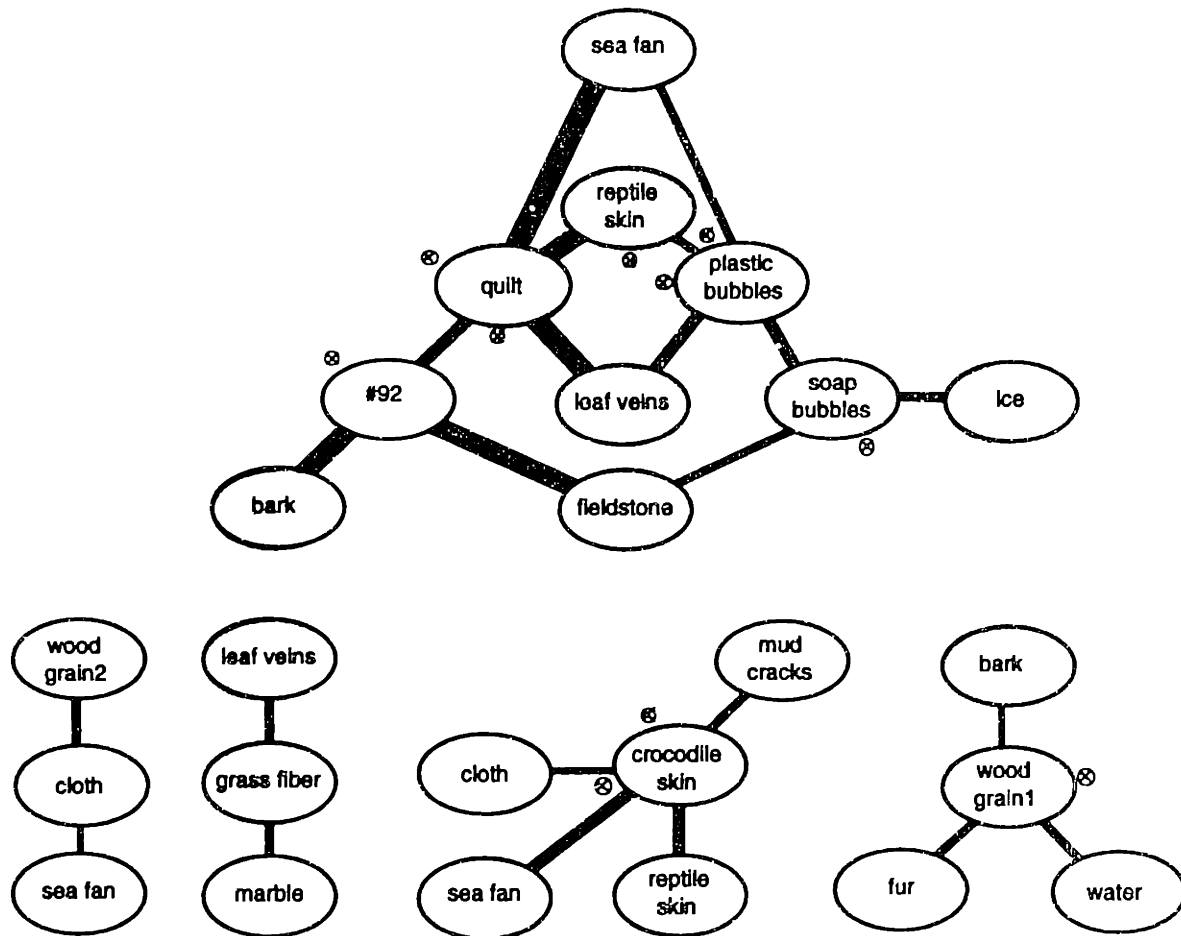


Figure 4: The separate trajectories of Figure 3.

fieldstone	2	woodgrain1	70	ice	100
reptile skin	3	soap bubbles	73	grass fiber	110
crocodile skin	10	cloth	76	plastic bubbles	111
bark	12	sea fan	87	leaf veins	137
water	37	#92	92	quilt	138
marble	63	fur	93	mud cracks	139
woodgrain2	68				

Table 2: A table of the corresponding stimuli numbers in Richards & Koenderink to our labels.

## References

Lokuge, I., Gilbert, S.A., & Richards, W. (1996) Structuring information with mental models: A tour of Boston. *Proceedings of ACM SIGCHI 96, Vancouver.*

Richards, W. & Koenderink, J. J. (1995) Trajectory mapping: A new nonmetric scaling technique. *Perception, 24* , 1315-1331.



## Appendix C

### Algorithm Code and Details

#### Part I: The Processing Narrative

We take the reader through a typical data processing session using as an example domain the kinship data examined in Chapter 4. We provide the commands used to go from subject data to a trajectory map. The software used in the process is Splus (a statistics and mathematics environment, Statistical Sciences), tma (my trajectory mapping algorithm), neato (graph-drawing algorithm courtesy of Stephen North, Bell Labs), and kyst (multidimensional-scaling software, Bell Labs). The basic stages of processing are:

1. input and parse a file of subject quintuples (Splus)
2. calculate triples in the data and write the file in a format for tma (Splus)
3. run the trajectory mapping algorithm (tma)
4. threshold the resulting graph and view it (neato)
5. run MDS on the adjacencies if desired (kyst)
6. calculate the diagnostics described in Chapter 3 (Splus)

The code for any customized Splus functions called in the series of commands below lies in Part II. The code for tma is available from the author.

In the narrative, all code and data files will be presented in `Courier` font. Splus commands will be preceded by `"Splus>"` and Unix commands preceded by `"Unix%".` Comments are often preceded by `"#"`. The ASCII text of data files will be bounded by `"-----"`.

Lastly, those unfamiliar with Splus will likely understand the gist of many commands given the knowledge that the underscore `"_"` serves as the assignment operator (what many people use `"="` for). The command below, for example, sets the variable `x` equal to 5. Those very familiar with Splus will notice that we leave some unimportant and simple-to-compute details of the data manipulation.

```
Splus> x_5
```

## 1. Collect subject data.

```
## 15 kin relations
```

```
Splus> kin.labels_c('grandfather','grandmother','grandson',  
  'granddaughter','brother','sister','father',  
  'mother','son','daughter','nephew','niece',  
  'uncle','aunt','cousin')
```

```
## Create a file for data entry that contains lines such as
```

```
Splus> makeTMtrials(1:15,'kin.quints')
```

```
## The file looks something like this, but contains all  
## possible pairs in random order.  If (3,12) is  
## included, pair (12,3) is *not*.
```

```
-----kin.quints  
  3   12  
 10    1  
  2    7  
  .  
  .  
  .  
-----
```

```
## After subject WR has completed the quintuplets, input the ## file.
```

```
Splus> wr.kin.data_scan('kinWR.quints')  
Splus> wr.kin.data_matrix(wr.kin.data,ncol=5,byrow=T)
```

## 2. Calculate triples and write them out.

```
## Calculate the triples within the quintuplets, as well  
## as the infeasible cases (nogos).
```

```
Splus> wr.kin.trips_triples(wr.kin,addDEs=1)  
Splus> wr.kin.nogos_getNoGos(wr.kin)
```

```
## Write the triples and nogos to a file that tma will  
## understand.
```

```
Splus> trip2ann(wr.kin.trips,wr.kin.nogos,file='kinWR.trip')
```

```
## The file looks something like this ("w 2" means that  
## triples with weight=2 follow):
```

```

-----kinWR.trip
# generated by Splus at Wed Oct 23 18:22:58 EDT 1996
w 2
 2 13 3
 1 14 6
 2 14 6
 6 5 8
 3 13 11
 4 13 11
 5 4 12
 8 4 12
 7 1 13
 9 2 13
 5 4 13
 5 6 14
 4 5 15
12 5 15
10 6 15
 2 9 15
w 3
 2 13 4
 4 8 5
 3 2 9
 8 4 10
 5 6 10
 7 1 11
 2 11 13
 6 5 15
w 4
 1 14 9
 9 2 14
 8 5 15
w 5
 4 10 7
 2 14 9
 5 8 12
 7 1 14
 5 6 15
w 6
 6 10 7
w 7
 9 2 11
w 8
 2 11 3
-----

```

### 3. Run tma, the trajectory mapping algorithm.

```

## Now feed the triples file to tma.
## Also give tma a parameters file that specifies how it will ## run. The 0 is a binary flag
the indicates whether or not
## file will be written out for the optimal graph at each
## setting of the cost function parameters.

```

```

Unix% tma tmpar 0 kinWR.trip

```

```
## Below is the tmpar file that we choose for all of our data
## collection after much experimentation. This is not to say
## it could not be improved.
```

```
-----tmpar
```

```
## simulated annealing and general parameters
```

```
oracle g                # metropolis alg. vs. gibbs sampling
initTemp 0.5           # initial temperature
annealFactor 0.9       # annealing factor at each temp. change
annealSteps 75         # max. number of annealing steps
maxTryFac 100          # max. number of tries at one temp.
maxChangeFac 10        # max states changes at one temp.
scaleWeights 1         # should triple weights be normalized?
nogoThreshFac .3       # factor used if wNoGos != 0 (below)
ignoreDummies 1        # should dummy nodes be ignored?
```

```
## costfunction parameters
```

```
## The syntax has 3 options:
```

```
## -- enter a single positive value to be used
## -- enter a series of values: -1 s [# of them] (# # ...)
## -- enter a range: -1 r [# of them] (min max)
```

```
wFailedTriples -1 r 10 (.1 5)
wBadSpacing 1
wFarSpacing -1 s 7 (.001 .005 .01 .1 1 2 3)
wMaxLinkNode 1
wTotalLinks -1 s 10 (.1 .25 .5 .75 1 2 3 4 5 6)
wDeadEnds 0
wNoGos 0
wSpurs 0
```

```
-----
## We sample the space of cost parameters at 700 points, and
## repeat each optimization twice, so that makes 1400
## optimizations.
```

```
## Note that we have not used tma to take dead ends or nogos
## (infeasibles) into account. That's a good goal for the
## next version.
```

```
## Running the algorithm produces several files:
```

```
##          kinWR.pDot
##          kinWR.gDot
##          kinWR.sp
##          kinWR.log
##          kinWR.avglinks
##          kinWR.maxlinks
##          kinWR.totallinks
```

```
## The first 3 files hold the same information in different
## formats: they describe links between nodes and the
## weights on those links. The *.pDot and *.gDot files are
## formatted for the neato graph-drawing software to read in
## and return PostScript and GIF output, respectively.
## The *.sp file is has no formatting tokens and is used
## by my program sp.
```

```
## Weights range from 0 - 10. Links with weights less than
## 1.0 are automatically deleted.
```

```
-----kinWR.pDot
graph G {
  size="7.5,10";
  ratio=compress;
  1 -- 7 [style="setlinewidth(2.34)"];
  1 -- 11 [style="setlinewidth(1.19)"];
  1 -- 13 [style="setlinewidth(1.21)"];
  1 -- 14 [style="setlinewidth(2.31)"];
  2 -- 3 [style="setlinewidth(1.29)"];
  2 -- 9 [style="setlinewidth(3.37)"];
  2 -- 11 [style="setlinewidth(3.69)"];
  2 -- 14 [style="setlinewidth(3.01)"];
  3 -- 11 [style="setlinewidth(3.71)"];
  4 -- 8 [style="setlinewidth(1.45)"];
  4 -- 10 [style="setlinewidth(3.26)"];
  5 -- 6 [style="setlinewidth(2.61)"];
  5 -- 8 [style="setlinewidth(2.56)"];
  5 -- 15 [style="setlinewidth(1.99)"];
  6 -- 10 [style="setlinewidth(3.39)"];
  6 -- 14 [style="setlinewidth(1.29)"];
  6 -- 15 [style="setlinewidth(2.64)"];
  7 -- 10 [style="setlinewidth(3.40)"];
  8 -- 12 [style="setlinewidth(2.67)"];
  9 -- 14 [style="setlinewidth(2.75)"];
}
```

```
-----
## The *.log file is a file that describes the parameters of
## each of the 1400 optimizations. The *links files are
## ascii files that contain statistics about the optimal
## graphs at each of the 1400 cost parameter settings.
## The first 3 columns are the cost parameters, and the
## last column is either the average links per node,
## the maximum number of links per node, or the total
## number of links in the optimal graph at that parameter
## setting.
```

```
-----kinWR.maxlinks
.
.
.
0.100000 3.911111 0.001000 4.000000
0.250000 3.911111 0.001000 3.500000
0.500000 3.911111 0.001000 3.500000
0.750000 3.911111 0.001000 3.000000
1.000000 3.911111 0.001000 3.500000
2.000000 3.911111 0.001000 3.000000
3.000000 3.911111 0.001000 2.000000
4.000000 3.911111 0.001000 0.000000
.
.
.
```

#### 4. Threshold the graph and view it.

```
## To view the graph, use neato software to convert data file
## to either GIF or PostScript, and then use either xv or
## ghostscript (shareware), respectively.
```

```
Unix% neato -Tgif kinWR.gDot | xv -
```

```
## or
```

```
Unix% neato -Tps kinWR.pDot > kinWR.ps
```

```
Unix% ghostview kinWR.ps
```

#### 5. Calculate adjacencies and run MDS.

```
## Build the adjacencies matrix, and write its lower
## triangle out in a format that will be friendly to kyst,
## the MDS algorithm.
```

```
Splus> wr.kin.adj_adjacencies(wr.kin)
```

```
Splus> writeLowerTri(wr.kin.adj, 'wr.kin.adj')
```

```
## Using emacs, insert wr.kin.adj into a data file
## wr.kin.kyst with data cards for kyst, the MDS program.
## The kyst manual comes with examples of data files.
```

#### 6. Calculate diagnostics on data and graphs.

```
##### Comparing alg and by-hand
```

```
## compare the by-hand graph and the algorithm graph
```

```
Splus> compareGraphs('kinWR.hand.sp', 'kinWR.sp')
```

```
$sim:
```

```
[1] 0.6817525
```

```
$numcomm:
```

```
[1] 16
```

```
$distinct1:
```

```
[1] 4
```

```
$distinct2:
```

```
[1] 4
```

```

## get the fit of the alg. graph to the triples

Splus> getFitMeasure('kinWR.spout',wr.kin.trips)
$rankcor:
[1] 0.6470588

$P.value:
[1] 0.01589629

$matched.trips:
[1] 16

$total.trips:
[1] 35

## get the fit of the by-hand graph to the triples

Splus> getFitMeasure('kinWR.hand.spout',wr.kin.trips)
$rankcor:
[1] 0.6070176

$P.value:
[1] 0.01206246

$matched.trips:
[1] 19

$total.trips:
[1] 35

## Calculate noise level in triples data.

Splus> wr.kin.comb_trip2comb(wr.kin.trips,15)
Splus> wr.kin.comb
[1] 1252  78  16  8  3  5  1  1  1

Splus> wr.kin.cd_wr.kin.comb/sum(wr.kin.comb)
Splus> wr.kin.cd
[1] 0.9172 0.0571 0.0117 0.0059 0.0022 0.0037 0.0007 0.0007 0.0007

Splus> rs15.400norm
[1] 0.8052 0.1775 0.0163 0.0009 2.3810e-05 0

N_wr.kin.cd[1:5]
n_rs15.400norm[1:5]

Splus> pchisq(sum( (N - n)^2/n ), 5)
[1] 0.002842245

```

## Part II: Code

Pieces of actual code that are referred to in the processing narrative. Subroutines that are included in a given piece of code are described immediately below.

```
##### Splus code

## "makeTMtrials" takes a vector and makes the randomly sorted
## pairs for filling in.

makeTMtrials_function(vec,filename)
{
  pairs_getPairs(vec)
  n_nrow(pairs)
  nh_floor(n/2)
  pairs_rand.sort(pairs)
  pairs_rbind(pairs[1:nh,],cbind(pairs[(nh+1):n,2],pairs[(nh+1):n,1]))
  pairs_rand.sort(pairs)

  x_cbind(rep(" ",n),
          pairs[,1],rep(" ",n),pairs[,2],rep(" ",n))

  write(format(t(x)),filename,ncol=5)
}

## "getPairs" takes a vector of numbers and returns all
## the possible pairs. If revflag=T, it includes
## pairs in both orders.

getPairs_function(x,revflag=F)
{
  n_length(x)
  base_n+1

  minfold_1*base + 2
  maxfold_n*base + n-1
  fold_seq(minfold,maxfold)
  num_length(fold)

  col1_floor(fold/base)
  col2_floor(fold - col1*base)
  allcol_cbind(col1,col2)

  bad_(1:num)[col1 == col2]
  if (!revflag) bad_c(bad,(1:num)[col2 < col1])

  bad_c(bad,(1:num)[col1 == 0])
  bad_c(bad,(1:num)[col2 == 0])

  matrix(x[allcol[-unique(bad),] ],ncol=2)
}
```



```

##### getting triples

# "triples" identifies the triples across rows within a matrix and
# gives frequency counts for them.
# from the original 5 column datarow, 1 2 3 4 5:
# method 0 is subsets, all possible ordered combos (10 triples)
# method 1 is adjacencies, ( (1,2,3), (2,3,4), (3,4,5) )
# method 2 is only the triples that the subject "considered":
# ( (1,2,4), (2,3,4), (2,4,5) )
# addDEs is "add dead ends"; if it's 1, then dead ends are
# converted to triples with new dummy nodes according to
# the specified triple method.
#
triples_function(data,thresh=2,matflag=F,method=1,addDEs=0)
(
  nr_nrow(data)
  nc_ncol(data)
  n_max(data)
  mat_array(0,c(n,n,n))
  for (i in c(1:nr)) {
    if (method==2) { trips_getTripsAB(data[i,]) }
    else {
      datarow_parseTMrow(data[i,])
      if (method==0) { trips_getSubsets(datarow,3) }
      if (method==1) { trips_getAdj(datarow,size=3) }
    }
    if (length(trips)) {
      for (j in c(1:nrow(trips))) {
        if (trips[j,1] > trips[j,3]) trips[j,]_rev(trips[j,])
        mat[ trips[j,1], trips[j,2], trips[j,3] ]_
          mat[ trips[j,1], trips[j,2], trips[j,3] ] + 1
      }
    }
  }
  res_tallyArray(mat,thresh)
  if (addDEs) {
    deadEnds_insertDummies(getDeadEnds(data,method))
    if (!is.null(deadEnds)) {
      res_unlist(list(res,deadEnds),recursive=F)
    }
  }
  if (matflag) list(res,matrix=mat)
  else res
)

### subroutines of triples

# "getTripsAB" takes a vector from TM and returns a NxSize matrix
# If the vector is E1 A I B E2, is returns
# E1 A B, A I B, and A B E2.
# It assumes -1 should be a "stop" flag and that a -2 should be
# skipped over.

```

```

getTripsAB_function(vec)
{
  res_numeric(0)
  if (vec[1] > 0) {
    res_rbind(res,cbind(vec[1],vec[2],vec[4]))
  }
  if (vec[3] > 0) {
    res_rbind(res,cbind(vec[2],vec[3],vec[4]))
  }
  if (vec[5] > 0) {
    res_rbind(res,cbind(vec[2],vec[4],vec[5]))
  }
  res
}

# "parseTMrow" takes a row of TM data and returns the row without
# -1's or -2's as appropriate.

parseTMrow_function(vec)
{
  if (vec[3]==-1) return(numeric())
  else vec[vec > 0]
}

# "getSubsets" takes a vector and returns the combinations that
# you would expect from ( n take m ) terminology. It includes A B
# but not B A, unless rev=T. It's recursive.

getSubsets_function(x,num=2,rev=F)
{
  if (num > length(x) || length(x)==0) return(numeric())
  r_numeric(0)
  if (num > 1) {
    for (i in (1:(length(x)-num+1)) ) {
      prev_cbind(x[i],getSubsets(x[-(1:i)],(num-1)) )
      r_rbind(r,prev)
    }
    num_num-1
  }
  else {
    for (i in (1:length(x)) ) {
      r_rbind(r,cbind(x[i]))
    }
  }
  if (rev) {
    r_rbind(r,r[, (ncol(r):1)])
  }
  else {
    r
  }
}

# "getAdj" takes a vector and returns a NxSize matrix in which
# each row is an adjacent subset of the vector of length "size".
# It assumes -1 should be a "stop" flag and that a -2 should be
# skipped over. It's used by the adjacencies and the triples
# function, among others.

```

```

getAdj_function(vec,size=2)
{
  n_length(vec)
  res_numeric(0)
  i_1
  while (i <= (n-size+1) ) {
    if (vec[i]== -1 || vec[i]== -2) {i_i+1; next }
    group_vec[i]
    j_0
    while (length(group) < size) {
      j_j+1
      if (i+j > n) { group_rep(-1,size); next }
      if (vec[i+j]== -1) { group_rep(-1,size); next }
      while (vec[i+j]== -2 & i+j <= n) j_j+1
      if (i+j > n) { group_rep(-1,size); next }
      group_c(group,vec[i+j])
    }
    if (group[1]== -1) {i_i+1; next }
    res_rbind(res,group)
    i_i+1
  }
  res
}

# "tallyArray" takes an array/matrix and, if the array is 3-D,
# for example, assumes that element (2,4,9)'s having a
# value of 5 means that the triplet 2,4,9
# occurred 5 times somewhere, and it returns a list of
# all triples that occurred [thresh] times, [thresh+1] times,
# etc. It assumes an NxNx... array of arbitrary dimensionality.

tallyArray_function(mat,thresh=2)
{
  L_length(mat)
  nn_max(mat)
  n_dim(mat)[1]
  res_vector("list", (nn-thresh+1))
  names(res)_paste("bin", (thresh:nn), sep='')
  for (i in c(thresh:nn)) {
    trip.list_(1:L)[mat==i]
    for (j in trip.list) {
      r_rev(unfold(j,n))
      r[-1]_r[-1]+1
      res[[i-thresh+1]]_rbind(res[[i-thresh+1]],r)
    }
  }
  res
}

# "getDeadEnds" gets the nodes which are dead ends and lists their
# penultimate nodes; if the data has -1 B C D E,
# method=1 takes C B -1, while method=2 takes D B -1, and
# method=0 takes both, as well as E B -1. Likewise in reverse.
#
# It builds a matrix where (x,x) has the number of times node
# x is a deadend. The other numbers in row x are the number of
# times that col y is a penultimate node for x.
#

```

```

getDeadEnds_function(data,method=2)
{
  maxd_max(data)
  data_data[data[,3] > 0,]
  nr_nrow(data)
  mat_matrix(0,nrow=maxd,ncol=maxd)
  for (i in (1:nr)) {
    if (data[i,1]== -1) {
      mat[data[i,2],data[i,2]]_mat[data[i,2],data[i,2]]+1
      if (method != 1) {
        mat[data[i,2],data[i,4]]_mat[data[i,2],data[i,4]]+1
      }
      if (method != 2) {
        mat[data[i,2],data[i,3]]_mat[data[i,2],data[i,3]]+1
      }
      if (method==0 & data[i,5] > 0) {
        mat[data[i,2],data[i,5]]_mat[data[i,2],data[i,5]]+1
      }
    }
    if (data[i,5]== -1) {
      mat[data[i,4],data[i,4]]_mat[data[i,4],data[i,4]]+1
      if (method != 1) {
        mat[data[i,4],data[i,2]]_mat[data[i,4],data[i,2]]+1
      }
      if (method != 2) {
        mat[data[i,4],data[i,3]]_mat[data[i,4],data[i,3]]+1
      }
      if (method==0 & data[i,1] > 0) {
        mat[data[i,4],data[i,1]]_mat[data[i,4],data[i,1]]+1
      }
    }
  }
  mat
}

# "insertDummies" takes a matrix from getDeadEnds and
# turns it into triples with dummy nodes with higher
# numbers for the dead ends. The weights are by
# frequency of the A B x triple.
#
# Note that if oneper=T, it pumps out pairs
# with one dummy per normal node. Plus, the
# weight of the pair = the number of dummies the
# normal node had.
#

```

```

insertDummies_function(demat,oneper=T)
{
  if (!(max(demat))) { return(NULL) }
  nr_nrow(demat)
  nextnode_nr+1
  nodiagmat_demat=diag(diag(demat))
  if (!oneper) maxbins_max(nodiagmat)
  else maxbins_max(demat)
  res_vector("list",maxbins+1)
  names(res)_c(1:maxbins,"dummies")
  for (i in (1:nr)) {
    if (demat[i,i]) {
      if (!oneper) {
        for (j in (1:nr)[nodiagmat[i,]>0]) {
          res[[demat[i,j]]]_rbind(res[[demat[i,j]]],c(i,j,nextnode))
          nextnode_nextnode+1
        }
      }
      else {
        res[[demat[i,i]]]_rbind(res[[demat[i,i]]],c(0,i,nextnode))
        nextnode_nextnode+1
      }
    }
  }
  res$dummies_c((nr+1),(nextnode-1))
  res
}

# "getNoGos" takes a matrix of quints (TM data) and
# returns a two column matrix of those rows that had
# -1 in the middle
getNoGos_function(data)
{
  data_data[data[,3]== -1,]
  if (!is.null(dim(data))) cbind(data[,2],data[,4])
}

# "trip2am" dumps triples out in format that tma
# (the sim annealing code) would like to read

```

```

trip2amn_function(liist,nogos=NULL,file)
{
  bins_names(liist)
  nc_3
  cat("# generated by Splus at",date(),"\n",file=file)
  for (i in (1:length(liist))) {
    if (length(liist[[i]]) & substring(bins[i],1,3)=="bin") {
      cat("w",substring(bins[i],4),"\n",file=file,append=T)
      write(format(t(liist[[i]])),file,ncol=nc,append=T)
    }
    if (bins[i]=="dummies") {
      cat("dummies",liist$dummies[1],"to",liist$dummies[2],"\n",
        file=file,append=T)
    }
    if (length(liist[[i]]) &
      !is.na(match(substring(bins[i],1,1),(0:9)))) {
      cat("dw",bins[i],"\n",file=file,append=T)
      write(format(t(liist[[i]])),file,ncol=nc,append=T)
    }
  }
  if (!is.null(nogos)) {
    cat("nogos\n",file=file,append=T)
    write(format(t(nogos)),file,ncol=2,append=T)
  }
  else cat("no nogos.\n")
}

# "adjacencies" counts adjacent pairs in quintuplet data.
# Plus, it deals with -1's and -2's. A -2 between
# two numbers is ignored, and they're counted as a neighboring
# pair. A -1 is a dead end. If the as.considered flag
# is false, it takes neighboring adjacencies, ie 1 2, 2 3, 3 4, 4 5.
# If it's true, it takes them based on the triples that the subjects
# considered, ie 1 2, 2 4, 2 3, 3 4, 2 4, 4 5.

adjacencies_function(data,as.considered=F)
{
  nr_nrow(data)
  nc_ncol(data)
  n_max(data)
  mat_matrix(0,nrow=n,ncol=n)
  for (i in c(1:nr)) {
    if (as.considered) { pairs_getAdjAB(data[i,]) }
    else { pairs_getAdj(data[i,]) }
    if (length(pairs)) {
      for (j in c(1:nrow(pairs))) {
        mat[ pairs[j,1], pairs[j,2] ]_
          mat[ pairs[j,1], pairs[j,2] ] + 1
      }
    }
  }
  mat
}

```

```

# "writeLowerTri" writes the lower triangle of a matrix to a file
writeLowerTri_function(mat,filename)
{
  n_ncol(mat)
  if (n != nrow(mat)) stop('ERROR: matrix must be square')

  vf_format(mat[2,])
  write(vf[1],filename,ncol=1)
  for (i in (3:n)) {
    vec_format(mat[i,])
    write(t(vec[1:(i-1)]),filename,ncol=i,append=T)
  }
}

##### diagnostics functions

## The goal here is build a measure of comparison across graphs

## Sim = (sum of common link weights - distinctive weights)
-----
          sum(weights in both graphs)

## if the graph data is in Splus, it's a either a square
## link-weight matrix or a 3-column matrix in the format of
## "FromNode ToNode Weight". This function assumes the latter.
## If you have the other, run mat2list. If you feed it a string,
## it'll try to load that sp file.
##
## This file currently assumes digraphs (bidirectional links)

compareGraphs_function(sp1,sp2)
{
  if (is.character(sp1)){
    sp1_scan(sp1)
    sp1_matrix(sp1,ncol=3,byrow=T)
  }
  if (is.character(sp2)){
    sp2_scan(sp2)
    sp2_matrix(sp2,ncol=3,byrow=T)
  }

  # normalize weights
  sp1[,3]_normalize(sp1[,3],.1,1)
  sp2[,3]_normalize(sp2[,3],.1,1)

  n_max(sp1[,1:2],sp2[,1:2])

  base_n+1

  allpairs_getPairs(1:n)
  allfold_base*allpairs[,1] + allpairs[,2]
  sp1fold_base*sp1[,1] + sp1[,2]
  sp1labels_match(sp1fold,allfold)
  sp2fold_base*sp2[,1] + sp2[,2]
  sp2labels_match(sp2fold,allfold)

```

```

comm.sp1_
  (1:length(sp1labels))[!is.na(match(sp1labels,sp2labels))]
comm.sp2_
  (1:length(sp2labels))[!is.na(match(sp2labels,sp1labels))]
num.comm_length(comm.sp1)
if (num.comm) {
  commW.sp1_sp1[comm.sp1,3]
  commW.sp2_sp2[comm.sp2,3]
  comm_sum((commW.sp1 + commW.sp2)/2)
  distinct_sum(sp1[-comm.sp1,3],sp2[-comm.sp2,3])
  # penalize for diffs in common weights
  distinct_distinct + sum(abs(commW.sp1-commW.sp2))
}
else {
  commW.sp1_0
  commW.sp2_0
  distinct_sum(sp1[,3],sp2[,3])
}
sim_(sum(commW.sp1,commW.sp2) - distinct)/sum(sp1[,3],sp2[,3])

list(sim=sim,numcomm=num.comm,distinct1=nrow(sp1)-num.comm,
      distinct2=nrow(sp2)-num.comm)
)

# start with a *.sp file
# then run sp on sp file to get a spout file.

getFitMeasure_function(spoutfile,trips)
(
  spfit_scan(spoutfile)
  spfit_matrix(spfit,ncol=5,byrow=T)
  n_max(spfit[,1:3])
  usedTrips_spfit[spfit[,5]==1,1:5]
  usedTrips_usedTrips[,1:4]
  usedTrips_usedTrips[order(usedTrips[,4]),]

  rtrips_rankTrips(trips)
  allTrips_makeTrips(1:n)

  usedTrips.lab_labelTrips(usedTrips,allTrips)
  usedRanks_usedTrips.lab[,1]

  rtrips.lab_labelTrips(rtrips,allTrips)
  dataRanks_rtrips.lab[,1]

  match.ud_match(usedRanks,dataRanks)
  match.ud_match.ud[!is.na(match.ud)]
  # In what order do the common rtrips occur in the usedTrips?
  # numbered according to rtrips.lab

  match.du_
  (1:length(dataRanks))[!is.na(match(dataRanks,usedRanks))]
  # the order that these same trips (common ones) appear in the
  # data, also numbered according to rtrips.lab

```



```

n2_length(match.du)
r_rankcor(match.du,match.ud)

# rankcor is distributed like student-t dist with n-2
# Numerical Recipes, p. 508
# pt() is the cumulative prob. of the t-dist.

r.t_r*(sqrt((n2-2)/1-r^2))
r.p_1 - pt(r.t,n2-2)

list(rankcor=r,p.value=r.p,matched.trips=n2,
      total.trips=nrow(xtrips))
}

#####

trip2comb_function(trips,n)
(
  comb_0
  nbins_length(trips)

  if (as.numeric(substring(names(trips)[1],4)) != 1)
    stop('triples data should include trips with weight 1.\n')

  for (i in 1:nbins) {
    if (is.null(nrow(trips[[i]]))) comb_c(comb,0)
    else comb_c(comb,nrow(trips[[i]]))
  }

  comb[1,_n]*(n-1)*(n-2)/2 - sum(comb[2:(nbins+1)])
  comb
)

```