Technical Report

UCAM-CL-TR-509
ISSN 1476-2986

Number 509

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Proximity visualisation of abstract data

Wojciech Basalaj

January 2001

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted October 2000 by the author for the degree of Doctor of Philosophy to the University of Cambridge.

ISSN 1476-2986

Abstract

Data Visualisation is an established technique for exploration, analysis, and presentation of data. A graphical representation is generated from the data content, and viewed by an observer, engaging vision – the human sense with the greatest bandwidth, and the ability to recognise patterns subconsciously. For instance, a correlation present between two variables can be elucidated with a scatter plot. An effective visualisation can be difficult to achieve for an abstract collection of objects, e.g. a database table with many attributes, or a set of multimedia documents, since there is no immediately obvious way of arranging the objects based on their content. Thankfully, similarity between pairs of elements of such a collection can be measured, and a good overview picture should respect this proximity information, by positioning similar objects close to one another, and far from dissimilar objects. The resulting *proximity visualisation* is a topology preserving map of the underlying data collection, and this work investigates various methods for generating such maps. A number of algorithms are devised, evaluated quantitatively by means of statistical inference, and qualitatively in a case study for each type of data collection. Other graphical representations for abstract data are surveyed, and compared to proximity visualisation.

A standard method for modelling proximity relations is Multidimensional Scaling (MDS) analysis. The result is usually a two- or three-dimensional configuration of points – each representing a single element from a collection, with inter-point distances approximating the corresponding proximities. The quality of this approximation can be expressed as a loss function, and the optimal arrangement can be found by minimising it numerically – a procedure known as least-squares metric MDS. This work presents a number of algorithmic instances of this problem, using established function optimisation heuristics: Newton-Raphson, Tabu Search, Genetic Algorithm, Iterative Majorization, and Simulated Annealing. Their effectiveness at minimising the loss function is measured for a representative sample of data collections, and the relative ranking is established. The popular classical scaling method serves as a benchmark for this study.

The computational cost of traditional MDS makes it unsuitable for visualising a large data collection. Incremental multidimensional scaling solves this problem by considering only a carefully chosen subset of all pairwise proximities. Elements that make up cluster diameters at a certain level of the single link cluster hierarchy are identified, and are subject to standard MDS, in order to establish the overall shape of the configuration. The remaining elements are positioned independently of one another with respect to this skeleton configuration. For very large collections the skeleton configuration can itself be built up incrementally. The incremental method is analysed for the compromise between solution quality and the proportion of proximities used, and compared to Principal Components Analysis on a number of large database tables.

In some applications it is convenient to represent individual objects by compact icons of fixed size, for example the use of thumbnails when visualising a set of images. Because the MDS analysis only takes the position of icons into account, and not their size, its direct use for visualisation may lead to partial or complete overlap of icons. Proximity Grid – an analogue of MDS in a discrete domain – is proposed to overcome this deficiency. Each element of an abstract data collection is represented within a single cell of the grid, and thus considerable detail can be shown without overlap. The proximity relationships are preserved by clustering similar elements in the grid, and keeping dissimilar ones apart. Algorithms for generating such an arrangement are presented, and compared in terms of output quality to one another, as well as standard MDS.

Acknowledgements

This work has been supported by an External Research Studentship received from Trinity College Cambridge, and by the Overseas Research Students Awards Scheme.

The author is indebted to Kerry Rodden for invaluable comments and ideas, and her help with statistical analysis and the production of figures with photograph arrangements.

Stephen Childs, Kerry Rodden, Peter Sewell, Pawel Wojciechowski, and Walt Yao contributed to the characterisation of algorithms presented in this dissertation, by permitting the author to run various computations on their workstations. Without their help the task could not have been completed on time.

A number of data sets have been used during the course of this work, and their originators are acknowledged in the appendices. Also, a number of existing algorithms served as an evaluation benchmark, and their authors or implementers are credited in the text, where the algorithms are first mentioned.

Last but not least, the author would like to thank his supervisor – Ken Moody – for providing a research environment that made this work possible.

Publications

Aspects of the work described in this dissertation feature in the following publications:

- W. Basalaj. Incremental multidimensional scaling method for database visualization. In *Proc. of Visual Data Exploration and Analysis VI*, SPIE volume 3643, pages 149–158, San Jose, California, January 1999.
- W. Basalaj and K. Eilbeck. Straight-line drawings of protein interactions. In *Proc. of Graph Drawing '99*, LNCS volume 1731, pages 259–266, Stirin, Czech Republic, September 1999.
- K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Evaluating a visualisation of image similarity as a tool for image browsing. In *Proc. of IEEE Information Visualization '99*, pages 36–43, San Francisco, October 1999.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is Proximity Visualisation

The focus of this dissertation is visualisation of abstract data collections, and an evaluation of a particular technique – *Proximity Visualisation* – suitable for the task. Database tables with many attributes, graphs, and multimedia collections are types of data collections for which it is difficult to design useful visual representations, since there is no immediate way to control the position of elements based on their content. However, similarity between elements of such collections can be measured, and a good overview picture should respect this proximity information, by positioning similar elements close to one another, and far from dissimilar ones. Such a visualisation is in effect a topology preserving map of the underlying data collection.

1.2 Data Types

The proximity of a pair of objects from a data collection can be expressed either as their similarity, mutual agreement, or dissimilarity – their distance in the abstract domain of the collection. The latter form is preferable for constructing a proximity visualisation of the collection, as distances between objects in the visual representation are to match the corresponding distances in the abstract domain. However, there exist transformations between the two variants of proximity measurements, and as long as at least one can be established for pairs of objects from a given data collection, its proximity visualisation can be formed.

A function that measures dissimilarity between a pair of objects from a collection is a dissimilarity coefficient. There may be a number of coefficients that could be used with a particular data collection, which in general will emphasise different aspects of object proximity. This section discusses suitable dissimilarity coefficients for a comprehensive range of types of abstract data collection, which reflect our experiences with applying proximity visualisation. It seems reasonable to assume that other types could be serviced by one of these dissimilarity coefficients, perhaps after certain reductions.

1.2.1 Quantitative

A quantitative variable can either be measured on an interval or a ratio measurement scale, for example temperature. If temperature is measured in degrees Celsius, i.e. an interval scale, the difference between two measurements t_1 and t_2 is meaningful, and so is the ratio of such differences: $(t_1 - t_3)/(t_2 - t_3)$. Additionally, a ratio scale, e.g. the Kelvin temperature scale, has a well defined zero point, and thus a ratio of two measurements t_1/t_2 becomes meaningful. Since the dissimilarity between two measurements can be defined in terms of their difference, both scales can be treated in the same way.

Suppose we have a set of n objects with the a^{th} attribute measured on the quantitative scale $\{u_{ia} : 1 \leq i \leq n\}$, and the range of this attribute is $R_a = \max_i u_{ia} - \min_i u_{ia}$. The dissimilarity δ_{ija} between a pair of objects (i, j) with regard to the a^{th} attribute can be defined as follows [Anderber73]:

$$\delta_{ija} = \frac{|u_{ia} - u_{ja}|}{R_a} \tag{1.1}$$

Dissimilarity coefficient (1.1) is always between 0 and 1, and can be viewed as a fractional disagreement between objects i and j on the a^{th} attribute, relative to the maximum disagreement given by R_a.

For quantitative measurement scales (and also those of Sections 1.2.2, 1.2.3, and 1.2.4) there exist many ways of combining dissimilarity scores on multiple attributes into an overall dissimilarity between a pair of objects [Anderber73]. However, we will not go into their details here, as an assumption that attributes are homogeneous, i.e. measured on the same scale, is very restrictive, and we prefer to treat this condition as a special case of heterogeneous data in Section 1.2.5.

1.2.2 Ordinal

An ordinal scale is weaker than quantitative, in that it also induces an ordering of objects, but does not make any statement about the magnitude of the differences. An example of an ordinal scale is the A–F system of grades; although there is a well defined ordering A > B > ... > F, it is impossible to say how the differences between pairs of grades relate, for instance whether the difference between A and B is greater than that between B and C. However, the difference between A and C must be greater than the difference between A and B, due to the inequalities, and an effective dissimilarity coefficient for a pair of objects measured on an ordinal scale can be based on differences in their rank.

Let a set of n objects with the a^{th} attribute measured on an ordinal scale be denoted as $\{u_{ia} : 1 \leq i \leq n\}$. If the rank order of an observation u_{ia} is given by $r(u_{ia})$, and the number of distinct ranks is $k_a = \max_i r(u_{ia})$, the dissimilarity between objects i and j on the a^{th} attribute takes the following form [Anderber73]:

$$\delta_{ija} = \frac{|r(u_{ia}) - r(u_{ja})|}{k_a - 1} \tag{1.2}$$

Coefficient (1.2) achieves in effect a transformation of an ordinal variable into a quantitative one, by replacing the ordinal measurements $\{u_{ia}\}$ with their rank order $\{r(u_{ia})\}$. Analogously to Section 1.2.1, (1.2) is also in the range $[0, 1]$, and represents the fractional disagreement between ranks of objects i and j on the a^{th} attribute, relative to the maximum observed rank difference.

1.2.3 Nominal

A nominal scale allows for the weakest form of measurement, as it does not enforce any ordering of objects. A number of non-overlapping categories are defined and an object can belong to a single one of these. The similarity of a pair of objects is defined in terms of their category membership: if they belong to the same category they are judged to be equivalent, otherwise they are completely different. An example of such an exclusive categorisation is a person's zodiac sign. For a set of n objects whose a^{th} attribute is measured on a nominal scale $\{u_{ia} : 1 \leq i \leq n\}$ pairwise dissimilarity between objects is defined as [Anderber73]:

$$\delta_{ija} = \begin{cases} 0 & : \quad u_{ia} = u_{ja} \\ 1 & : \quad \text{otherwise} \end{cases} \tag{1.3}$$

1.2.4 Binary

A binary scale of measurement is a special case of a nominal scale with only two categories. If both categories have the same significance, for example male–female, (1.3) can be used as the dissimilarity coefficient for such an attribute. However, if one of the categories denotes an absence of some property, e.g. yes–no, on–off, a negative match of a pair of objects is not considered as significant as a positive match. The dissimilarity between a pair of objects from the set $\{u_{ia} : 1 \leq i \leq n\}$, with the a^{th} attribute measured on a binary scale, then becomes [Anderber73]:

$$\delta_{ija} = \begin{cases} 0 & : \quad u_{ia} = u_{ja} = 1 \\ 1 & : \quad \text{otherwise} \end{cases} \tag{1.4}$$

In creating ontologies entities are scored on a number of binary attributes, and typically only a small proportion of them will be positive for any given entity. In such cases it might be advantageous to treat an attribute on which both entities scored negative as missing (see Section 1.2.5), so as not to unduly bias the judgement of their pairwise similarity [Gower71].

1.2.5 Heterogeneous

A collection of entity descriptions may be conveniently represented by a set of tuples or a set of objects with appropriate attributes. The utility of relational and object databases is based on this premise. The canonical representation of such data is a table with one row for each object, and a column for each attribute – Table 2.1 on page 12 provides an example. In general, the attributes cannot be expected to be homogeneous, and thus a dissimilarity coefficient for data tables has to combine attributes measured on arbitrary scales, to give an overall dissimilarity between pairs of objects (rows).

A general similarity coefficient [Gower71] is a suitable coefficient for heterogeneous data, and can accommodate missing values and conditional attributes – a crucial ability if any possible real world data is to be considered. An analogous general dissimilarity coefficient can be defined for a set of n objects with q attributes $\{u_{ia} : 1 \leq i \leq n, 1 \leq a \leq q\}$ as follows:

$$\delta_{ij} = \frac{\sum\limits_{a=1}^{q} w_{ija} w_a \delta_{ija}}{\sum\limits_{a=1}^{q} w_{ija} w_a} \tag{1.5}$$

where w_{ija} takes value 1 if objects i and j can be compared on the a^{th} attribute, and 0 otherwise; w_a is the weight given to attribute a; δ_{ija} is the dissimilarity between objects i and j as measured on the a^{th} attribute according to (1.1), (1.2), (1.3), or (1.4) depending on the type of this attribute.

Each δ_{ija} provides the distance in a uniform scale between a given pair of objects (i, j) with respect to a single attribute a. The general dissimilarity coefficient is a framework for combining individual distances for all attributes into an overall distance in a uniform representation space. It can be seen that a weighted City block metric, normalised by the maximum distance, is used in effect:

$$D_1(\boldsymbol{\delta}^{(ij)}) = \sum_{a=1}^{q} w_a \delta_{ija} \tag{1.6}$$

where $\boldsymbol{\delta}^{(ij)} = (\delta_{ij1}, \ldots, \delta_{ijq})^{T}$[†] is a vector of distances for individual axes. Therefore, the general dissimilarity coefficient can be generalised to any instance of the weighted Minkowski metric[‡]:

$$D_\lambda(\boldsymbol{\delta}^{(ij)}) = \left(\sum_{a=1}^{q} w_a \delta_{ija}^{\lambda}\right)^{\frac{1}{\lambda}}, \ \lambda \geq 1 \tag{1.7}$$

to give the λ-general dissimilarity coefficient:

$$\delta_{ij}^{(\lambda)} = \left(\frac{\sum\limits_{a=1}^{q} w_{ija} w_a \delta_{ija}^{\lambda}}{\sum\limits_{a=1}^{q} w_{ija} w_a}\right)^{\frac{1}{\lambda}} \tag{1.8}$$

For $\lambda = 2$ and purely quantitative attributes (1.8) simplifies to the weighted Euclidean distance normalised to the range $[0, 1]$, and for this desirable property we decided to use this form of the λ-general dissimilarity coefficient for data tables in the sequel.

[†]henceforth, all vectors are taken as column vectors; a row vector can be obtained from the column vector, and vice versa, by transposition, which is denoted with the T superscript

[‡]D_2 corresponds to the Euclidean distance, and D_∞ is equivalent to $\max\limits_{a} \delta_{ija}$

1.2.6 Relationships

Binary relationships $x \, R \, y$ within a set V of entities can be expressed as a graph $G(V, E = \{(a, b) : a, b \in V, a \, R \, b\})$, with an edge (a, b) whenever entities a and b enter into a relationship, and an entity being synonymous to a vertex. In general, a pair of entities might enter into an indirect relationship through one or more intermediate entities. Such a chain of vertices defines a path through the graph G that has a length equal to the number of chained vertices less one, i.e. the number of edges that have to be traversed.

The graph theoretic distance l is a function that for a given pair of vertices $a, b \in V$ returns the length of the shortest path between them. For undirected graphs l satisfies metric properties:

1. non-negativity: $l(a, b) \geq 0$, for all $a, b \in V$
2. identity: $l(a, b) = 0$, if and only if $a = b$
3. symmetry: $l(a, b) = l(b, a)$, for all $a, b \in V$
4. triangle inequality: $l(a, b) \leq l(a, c) + l(c, b)$, for all $a, b, c \in V$. If c is a vertex on the shortest path between a and b then $l(a, b) = l(a, c) + l(c, b)$. Otherwise, $l(a, b) > l(a, c) + l(c, b)$ cannot hold because a path through c would be shorter.

These properties are preserved for a disconnected graph if the length of the path between vertices from separate components is considered to be infinite, though for practical purposes a value greater than the longest path in all of the connected components will suffice. Thus, l is a suitable and intuitive dissimilarity coefficient for undirected graphs [Kruskal78a].

Directed graphs can be accommodated by treating all edges as undirected for the purposes of calculating dissimilarity between vertices, and taking account of the direction of edges only in their visual representation. Alternatively, the third metric property could be dispensed with, making dissimilarities asymmetric. If the strength of relationships can vary appropriate weights can be assigned to edges, and l generalised to return the path with the least cumulative weight for a given pair of vertices.

1.2.7 Images

Humans are adept at determining similarity between a given pair of images, by utilising the capabilities of the visual cortex to recognise objects in both images, and establishing semantic relationships between these objects. This process is difficult to emulate on a computer system, with the present knowledge of neurology and artificial learning. However, it is practical to automatically extract low-level features from the images, such as colour and texture, and compare their similarity to give an overall similarity between the two images. In our work we have been using IRIS (the acronym stands for Image Regions In Summary) – an image similarity coefficient developed at AT&T Laboratories in Cambridge, which combines global image properties in the form of colour histograms and local, region based features [Rodden99a].

During a pre-processing step an image is segmented into regions of broadly homogeneous colour properties. Regions are then classified as either large or small. The large regions are further classified as either textured or smooth, and the small regions as regularly or irregularly shaped. The image is partitioned into nine areas in a 3 by 3 grid. Subsequently, a colour histogram for each of the 4 types of region is recorded, and the largest (dominant) region is identified, in an area summary. The dissimilarity between a pair of images is computed from their summaries: the χ^2 statistic is used to determine the distance between histograms from corresponding areas, the Mahalanobis distance in RGB colour space is used for dominant regions, and the final coefficient is a weighted sum of these distances.

IRIS has been designed specifically to perform well on indexing and searching of image databases. There exist a multitude of alternatives, contributed by the Image Retrieval community, for calculating low-level visual similarity between pairs of images. These coefficients are of varied complexity and retrieval performance, however they seem to be roughly equivalent for the purposes of visualisation [Rodden00].

1.2.8 Text Corpus

It is common practice to index a corpus of documents on themes occurring within it, e.g. nouns or phrases, to facilitate querying for relevant themes. In a classic model of Information Retrieval – the vector model – each document is represented by a weight vector $\boldsymbol{w} = (w_1, \ldots, w_q)^T$, where element w_a specifies the relative importance of theme t_a in the document, and q is the total number of themes and the dimensionality of the weight vector space [Baeza-Ya99]. If the theme t_a does not occur in the document, the element w_a is simply set to 0, otherwise w_a is made proportional to the frequency of t_a's occurrence in the document, and inversely proportional to the commonness of t_a within the corpus. Thus, a theme frequently occurring in a given document will be assumed to have high relevance, unless it is common to all the documents, e.g. 'computer' in a corpus of computer science articles.

A measure of similarity s_{ij} between documents i and j, or equivalently vectors $\boldsymbol{w}^{(i)}$ and $\boldsymbol{w}^{(j)}$ in the vector model, is a cosine of the angle between $\boldsymbol{w}^{(i)}$ and $\boldsymbol{w}^{(j)}$:

$$s_{ij} = \frac{\boldsymbol{w}^{(i)T}\boldsymbol{w}^{(j)}}{\|\boldsymbol{w}^{(i)}\|\|\boldsymbol{w}^{(j)}\|} = \frac{\sum\limits_{a=1}^{q} w_a^{(i)} w_a^{(j)}}{\left(\sum\limits_{a=1}^{q} w_a^{(i)2}\right)^{\frac{1}{2}} \left(\sum\limits_{a=1}^{q} w_a^{(j)2}\right)^{\frac{1}{2}}} \tag{1.9}$$

where $\|\boldsymbol{w}\|$ denotes the Euclidean norm (length) of the vector $\boldsymbol{w}$. Since theme weights are non-negative, s_{ij} will be in the range $[0, 1]$, and to convert it to a dissimilarity δ_{ij} between documents i and j, the following formula can be used:

$$\delta_{ij} = \sqrt{1 - s_{ij}} \tag{1.10}$$

Let $\boldsymbol{V} = \left(\boldsymbol{w}^{(1)}/\|\boldsymbol{w}^{(1)}\|, \ldots, \boldsymbol{w}^{(n)}/\|\boldsymbol{w}^{(n)}\|\right)^T$ be the matrix of normalised weights for all n documents in the corpus; the complete similarity matrix can be expressed as $\boldsymbol{S} = [s_{ij}] = \boldsymbol{V}\boldsymbol{V}^T$, and is positive semi-definite by definition. Consequently, the dissimilarity matrix calculated by applying (1.10) to every element of $\boldsymbol{S}$ is Euclidean[§] [Gower86] – a desirable property for its visual representation.

1.2.9 Proximity

There are data collections that consist solely of proximity measurements, for example results of an experiment where a subject has been asked to rate how similar pairs of stimuli are. The only meaningful visual representation for such data is one based on proximity. If similarity measurements have actually been collected they have to be transformed to the corresponding dissimilarity scores, prior to constructing their visualisation, by using (1.10) for instance. Additionally, unstandardised similarities have to be scaled by the largest similarity to bring them to the range $[0, 1]$, which is a precondition for the transformation (1.10).

Since judgements of dissimilarity have been collected directly, they are unlikely to obey Euclidean or even metric properties – most importantly triangle inequality – which will make their proximity visualisation of dubious quality. However, by adding a sufficiently large positive constant to every dissimilarity, except self-dissimilarities which should always be 0, they can be made metric and Euclidean [Gower86, Borg97].

The magnitude of dissimilarity judgements may be deemed not to be reliable, and only their relative order, exactly the distinction between a quantitative scale (see Section 1.2.1) and an ordinal scale (see Section 1.2.2). In this case the dissimilarities should be replaced by their rank order. Such a procedure will cancel the effect of outlying dissimilarity measurements, which could otherwise greatly distort the visual representation.

1.3 Dissertation Aims

For the types of data discussed in Section 1.2 there exist many visualisation techniques; however, they are applicable to just a single data type, in general. On the other hand, proximity visualisation is generic, and can be used with any data collection, as long as a suitable dissimilarity coefficient can be defined for it. Analysis of quantitative data is the main concern of Statistics and Data Visualisation, and thus the greatest number of visualisation methods target collections of objects

[§]an $n \times n$ matrix $[d_{ij}]$ is said to be Euclidean if n points can be embedded in a Euclidean space of some dimension, such that the Euclidean distance between the ith and jth points is d_{ij}, for all $1 \leq i, j \leq n$

measured on multiple quantitative variables. Chapter 2 presents an overview of multivariate visualisation, and compares the most influential techniques.

An established technique for creating proximity visualisations is introduced in Chapter 2, and its detailed account is the purpose of Chapter 3. The underlying theory is presented first, followed by examples of practical algorithms for meeting that specification. To put the discussion on a more rigorous footing, these algorithms are subsequently evaluated based on the quality of proximity visualisations they supply, and their responsiveness.

The algorithms of Chapter 3 are unsuitable for very large data collections, as they will require an inordinate amount of time for computing the corresponding proximity visualisations. Chapter 4 discusses the use of cluster analysis to pre-process the data, so that the most important objects from a collection are identified. The proximity of such marker objects is visualised exactly; the remaining objects are located by reference to these markers only. Thus, a way of achieving a compromise between quality and responsiveness is established.

Chapter 5 takes a different approach to proximity visualisation. The emphasis is not on preserving dissimilarities as well as possible, or coping with large data collections, but with achieving a display with high information density. Such a display allows the use of complex icons to augment the visual representation of proximity, useful for browsing a multimedia collection or searching for a specific element from it, for example. This scenario is further explored in Chapter 6.

A browsing interface for a collection of images, based on proximity visualisations using both visual proximity of images and proximity of image captions, is just one of the case studies presented in Chapter 6. Interactions between proteins in a cell can be considered as an undirected graph, and an example protein graph drawing is studied in comparison to an equivalent textbook diagram. The final case study is that of database visualisation, where a unified view of data and metadata can be achieved by means of proximity visualisation.

The results of this dissertation are summarised in Chapter 7, and some concluding remarks are offered.

1.4 Evaluation Framework

Chapters 3, 4, and 5 each present a number of alternatives for creating proximity visualisations. Given a visual representation, it is possible to assess objectively how well dissimilarities among all pairs of objects from a collection are preserved. Thus, every algorithm is exercised with a test bed of data collections, and an objective measurement of quality is taken for each test case. Subsequently, statistical analysis is applied to rank the algorithms by their overall quality. An analogous analysis is performed with respect to the time taken to compute proximity visualisations with each algorithm, and conclusions are drawn from the trade-off between these two criteria.

It is important to complement the statistical analysis with a subjective assessment of proximity visualisations produced with different algorithms. It is conceivable to imagine that algorithms can be comparable in terms of objective quality, but deliver proximity visualisations with different distribution of objects, or other disparate characteristics. Consequently, examining these visualisations might establish that the output of a certain algorithm is more pleasing or intuitive, and thus favour it over others. For algorithms that differ significantly on objective quality it is still advantageous to compare visualisations generated by them, to find out the character of the differences.

Chapter 2

Multivariate Visualisation Techniques

Visual exploration of multivariate data is of great interest in Statistics and Information Visualisation. A number of methods have been proposed in both fields, ranging from the very useful to the quirky. This chapter introduces a few of the most established multivariate visualisation techniques by example. The criterion for selection was generality, and suitability for the non-interactive and flat medium of paper. The effectiveness of the methods is compared, and evaluated relative to their limitations.

2.1 Running Example

Several multivariate visualisation techniques have been presented with a challenge in the form of the *cars* data table [Henderso81], which is reproduced in Table 2.1, and is also mentioned in Section B.1 of the appendices. This data table contains a record of 38 cars manufactured in the period 1978–79, with the following attributes:

1. primary country of the manufacturer
2. model name
3. miles per gallon – a measure of petrol efficiency assessed on the race track
4. weight in thousands of lbs
5. drive ratio in the highest gear
6. horsepower
7. engine displacement in cubic inches
8. number of cylinders

Table 2.1: Details of the *cars* data table

country	model name	mpg	weight	ratio	hp	disp.	cyl.
USA	Buick Estate Wagon	16.9	4.360	2.73	155	350	8
USA	Ford Country Squire Wagon	15.5	4.054	2.26	142	351	8
USA	Chevy Malibu Wagon	19.2	3.605	2.56	125	267	8
USA	Chrysler LeBaron Wagon	18.5	3.940	2.45	150	360	8
USA	Chevette	30.0	2.155	3.70	68	98	4
Japan	Toyota Corona	27.5	2.560	3.05	95	134	4
Japan	Datsun 510	27.2	2.300	3.54	97	119	4
USA	Dodge Omni	30.9	2.230	3.37	75	105	4
Germany	Audi 5000	20.3	2.830	3.90	103	131	5
Sweden	Volvo 240 GL	17.0	3.140	3.50	125	163	6
Sweden	Saab 99 GLE	21.6	2.795	3.77	115	121	4
France	Peugeot 694 SL	16.2	3.410	3.58	133	163	6
USA	Buick Century Special	20.6	3.380	2.73	105	231	6
USA	Mercury Zephyr	20.8	3.070	3.08	85	200	6
USA	Dodge Aspen	18.6	3.620	2.71	110	225	6
USA	AMC Concord D/L	18.1	3.410	2.73	120	258	6
USA	Chevy Caprice Classic	17.0	3.840	2.41	130	305	8
USA	Ford LTD	17.6	3.725	2.26	129	302	8
USA	Mercury Grand Marquis	16.5	3.955	2.26	138	351	8
USA	Dodge St Regis	18.2	3.830	2.45	135	318	8
USA	Ford Mustang 4	26.5	2.585	3.08	88	140	4
USA	Ford Mustang Ghia	21.9	2.910	3.08	109	171	6
Japan	Mazda GLC	34.1	1.975	3.73	65	86	4
Japan	Dodge Colt	35.1	1.915	2.97	80	98	4
USA	AMC Spirit	27.4	2.670	3.08	80	121	4
Germany	VW Scirocco	31.5	1.990	3.78	71	89	4
Japan	Honda Accord LX	29.5	2.135	3.05	68	98	4
USA	Buick Skylark	28.4	2.670	2.53	90	151	4
USA	Chevy Citation	28.8	2.595	2.69	115	173	6
USA	Olds Omega	26.8	2.700	2.84	115	173	6
USA	Pontiac Phoenix	33.5	2.556	2.69	90	151	4
USA	Plymouth Horizon	34.2	2.200	3.37	70	105	4
Japan	Datsun 210	31.8	2.020	3.70	65	85	4
Italy	Fiat Strada	37.3	2.130	3.10	69	91	4
Germany	VW Dasher	30.5	2.190	3.70	78	97	4
Japan	Datsun 810	22.0	2.815	3.70	97	146	6
Germany	BMW 320i	21.5	2.600	3.64	110	121	4
Germany	VW Rabbit	31.9	1.925	3.78	71	89	4

The first attribute is measured on a nominal scale (see Section 1.2.3), the second is a label (see Section 1.2.8), the remaining attributes are quantitative (see Section 1.2.1). The task set out for each visualisation method is that of bringing out the differences and similarities between cars on the basis of their drive parameters. We felt that including the first two attributes would prejudice this analysis, and cause cars from the same manufacturer or just the same country to appear more similar.

2.2 Parallel Coordinates

A single row $\boldsymbol{u}_i^T = (u_{i1}, \ldots, u_{iq})$ of a data table with q attributes, measured on any scale apart from nominal (see Section 1.2.3), can be thought of as a point in a q-dimensional Cartesian coordinate system, with the abscissa on the a^{th} axis given by u_{ia}. For $q > 3$ such configurations of points cannot be directly visualised; the method of Parallel Coordinates overcomes this limitation by arranging axes vertically, and spacing them uniformly across the plane [Inselber85]. Point $\boldsymbol{u}_i$ in this coordinate system is a polygonal line connecting the corresponding abscissas on the parallel axes.

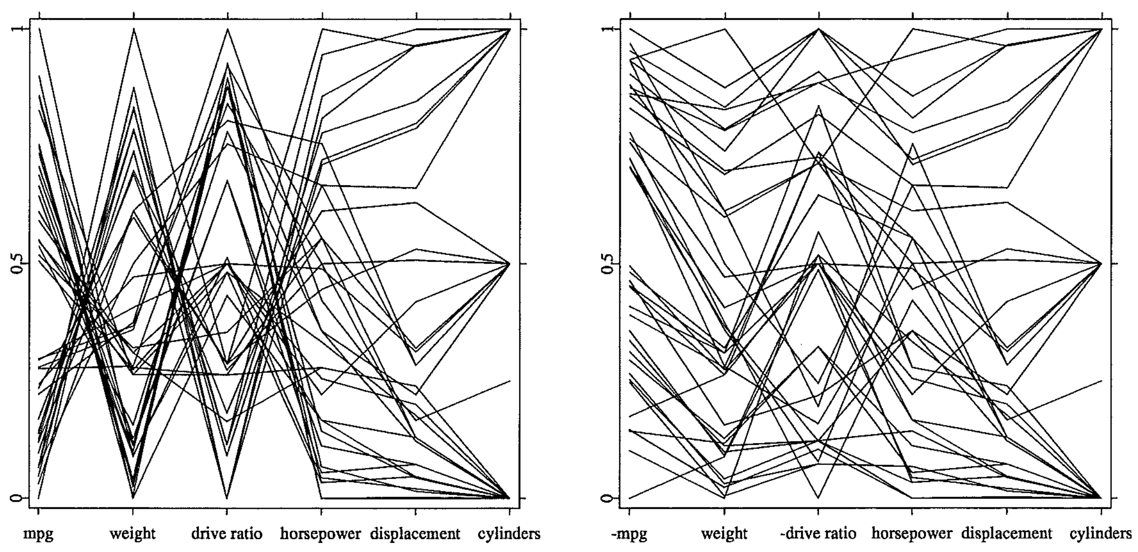
It is apparent from the parallel coordinates visualisation of the *cars* data table in Figure 2.1(a) that the last three attributes are substantially correlated. Moreover, lines representing the individual rows cross over between mpg, weight, drive ratio, and horsepower attributes, suggesting that these attributes might be negatively correlated in pairs. Inverting the mpg and drive ratio axes leads to a much clearer visualisation in Figure 2.1(b), which could be improved further by permuting the order of axes. The need for such a high level of customisation presents the ultimate obstacle in effectively visualising many variables with this method, made worse if the number of observations, and hence lines, is large.

2.3 Andrews Plot

In an Andrews plot each row $\boldsymbol{u}_i^T = (u_{i1}, \ldots, u_{iq})$ of a data table with q attributes is represented by a line, similarly to parallel coordinates. In this case it is a curve defined by the following trigonometric function [Andrews72]:

$$f_{u_i}(t) = \underbrace{\frac{u_{i1}}{\sqrt{2}} + u_{i2}\sin(t) + u_{i3}\cos(t) + u_{i4}sin(2t) + u_{i5}cos(2t) + \ldots}_{q} \tag{2.1}$$

plotted over the interval $t \in (-\pi, \pi)$. It is recommended that the most important attributes are associated with the low frequency terms, as they determine the overall shape of the curve. This might entail an iterative and exploratory approach to determine a satisfactory assignment, in the same way as for parallel coordinates.



(a) original (b) correlated

Figure 2.1: Parallel coordinates

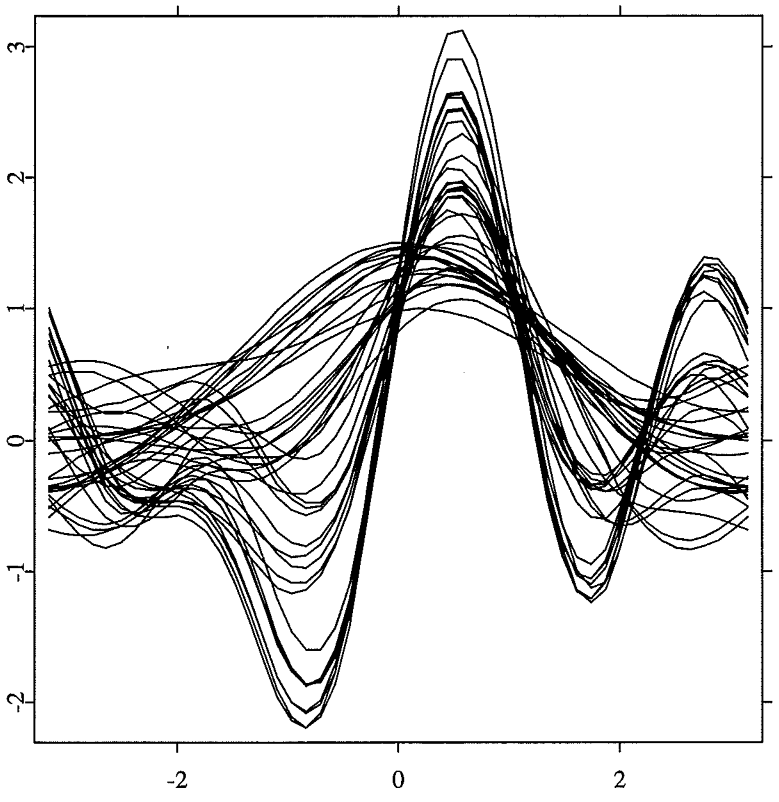


Figure 2.2: Andrews plot

Let $\bar{\boldsymbol{u}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{u}_i$ denote the mean of the n rows $\boldsymbol{u}_i^T$ of the data table; function (2.1) preserves this mean:

$$f_{\bar{\boldsymbol{u}}}(t) = \frac{1}{n}\sum_{i=1}^{n} f_{\boldsymbol{u}_i}(t) \tag{2.2}$$

so that the plot of $\bar{\boldsymbol{u}}$ is a pointwise average of the plots for individual rows. Another useful property of (2.1) is that it preserves the Euclidean distance $\|\boldsymbol{u}_i - \boldsymbol{u}_j\|$ between pairs of points in the q-dimensional space:

$$\int_{-\pi}^{\pi} \left(f_{\boldsymbol{u}_i}(t) - f_{\boldsymbol{u}_j}(t)\right)^2 dt = \pi\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2 = \pi\sum_{a=1}^{q} (u_{ia} - u_{ja})^2 \tag{2.3}$$

Thus, close points will result in similar plots, and plots for distant points will be distinct. These features are useful for detecting clusters and outliers, and are common to the parallel coordinates technique. Andrews plots have a number of other characteristics, especially helpful in statistical analysis of the underlying data [Andrews72].

Figure 2.2 is an Andrews plot of the *cars* data table. There seem to be two extreme clusters of cars. The remaining observations fall between the extremes, and form a loose cluster, which can be separated from the first two at $t = -1$ and $t = 2$. Additional insight could be gained by plotting these clusters separately, and in fact it is recommended that no more than 10 points $\boldsymbol{u}_i$ are plotted at a time for a detailed examination [Andrews72].

2.4 Multidimensional Scaling

Like parallel coordinates and Andrews plots, Multidimensional Scaling can also be used to visualise multivariate data [Borg97, Cox94]. However, the original q axes and coordinates of points $\boldsymbol{u}_i = (u_{i1}, \ldots, u_{iq})^T$ do not enter the visualisation directly. Instead, a configuration of points $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$ is found in a space of lower dimension $p < q$, such that all inter-point distances $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ match as closely as possible the original distances $\|\boldsymbol{u}_i - \boldsymbol{u}_j\|$. A two- or three-dimensional embedding is an obvious choice for visualisation; higher values of p can be useful for statistical analysis. A more elaborate description of this method is presented in Chapter 3.

It might be helpful to envisage the process of multidimensional scaling in two dimensions as wrapping a surface – an elastic sheet – around points $\{\boldsymbol{u}_i\}$ in the original high dimensional space, and taking $\boldsymbol{x}_i$ as the projection of $\boldsymbol{u}_i$ onto this surface. In effect a non-linear mapping between the two configurations is established, and it is likely to be superior for purposes of visualisation to rotating a rigid plane in the high dimensional space to find the closest fit to $\{\boldsymbol{u}_i\}$, a procedure known as Principal Components Analysis [Pearson01].

A two-dimensional multidimensional scaling configuration for the *cars* data set is presented in Figure 2.3. Inspection of the corresponding Andrews plot in Section 2.3 led to the conclusion that there are three clusters of cars. These

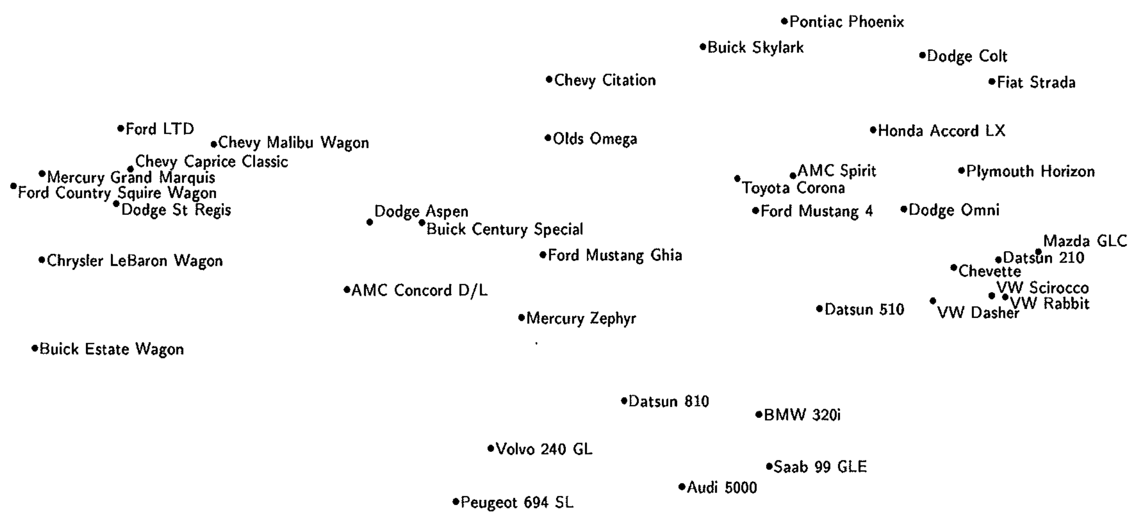


Figure 2.3: Multidimensional scaling

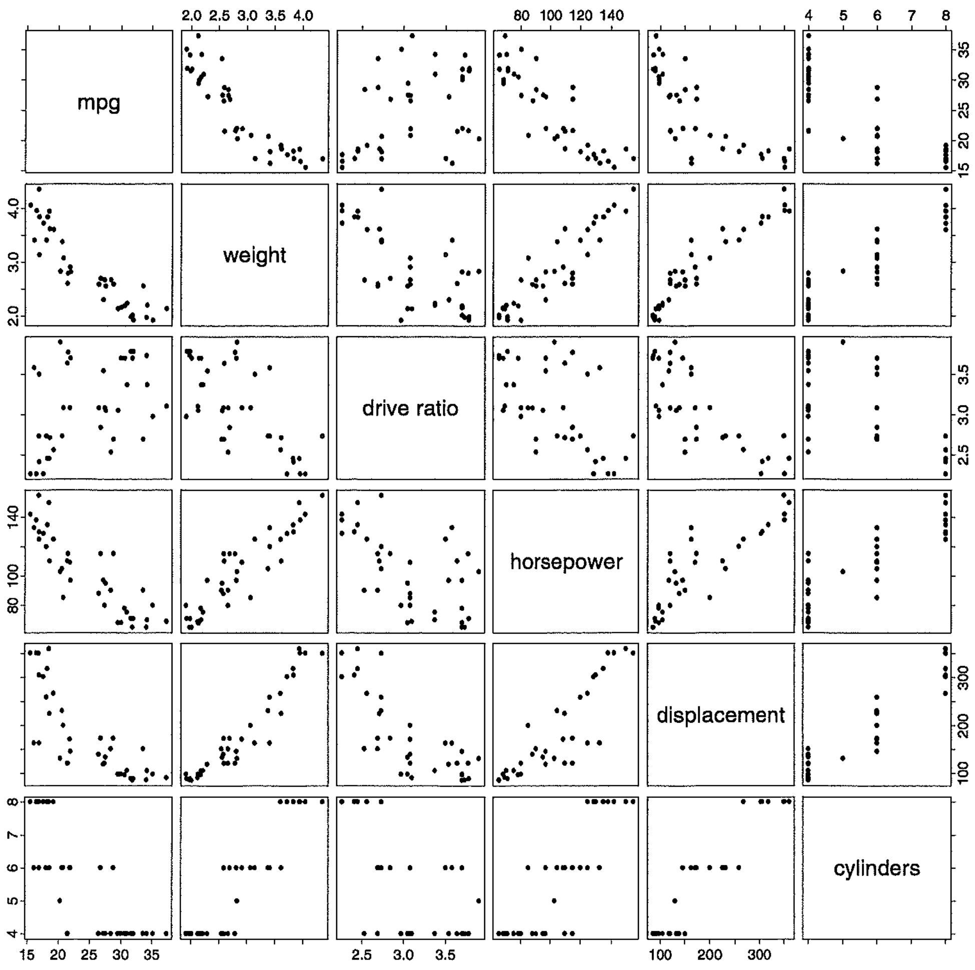


Figure 2.4: Scatterplot matrix

clusters are apparent from Figure 2.3, and can be readily verified to group cars with 8 cylinders on the left hand side of the figure, 6 and 5 in the middle, and 4 on the right. In effect a map is constructed that charts individual cars based on the overall similarity of their drive parameters – a Proximity Visualisation, in other words.

2.5 Scatterplot Matrix

A scatterplot matrix is a collection of scatterplots organised analogously to a co-variance matrix, with variable a plotted against variable b in the a^{th} row and b^{th} column of the matrix [Clevelan84]. The diagonal plots can show the distribution of individual variables, or simply be placeholders for variable names, as is the case for the scatterplot matrix representation of the *cars* data table in Figure 2.4. Individual scatterplots can reveal correlations between variables, for example linearity, and the complete matrix can be useful for an initial exploration of a data set. However, the display becomes overwhelming with anything more than a few variables; lack of a unified representation of data is also a serious drawback.

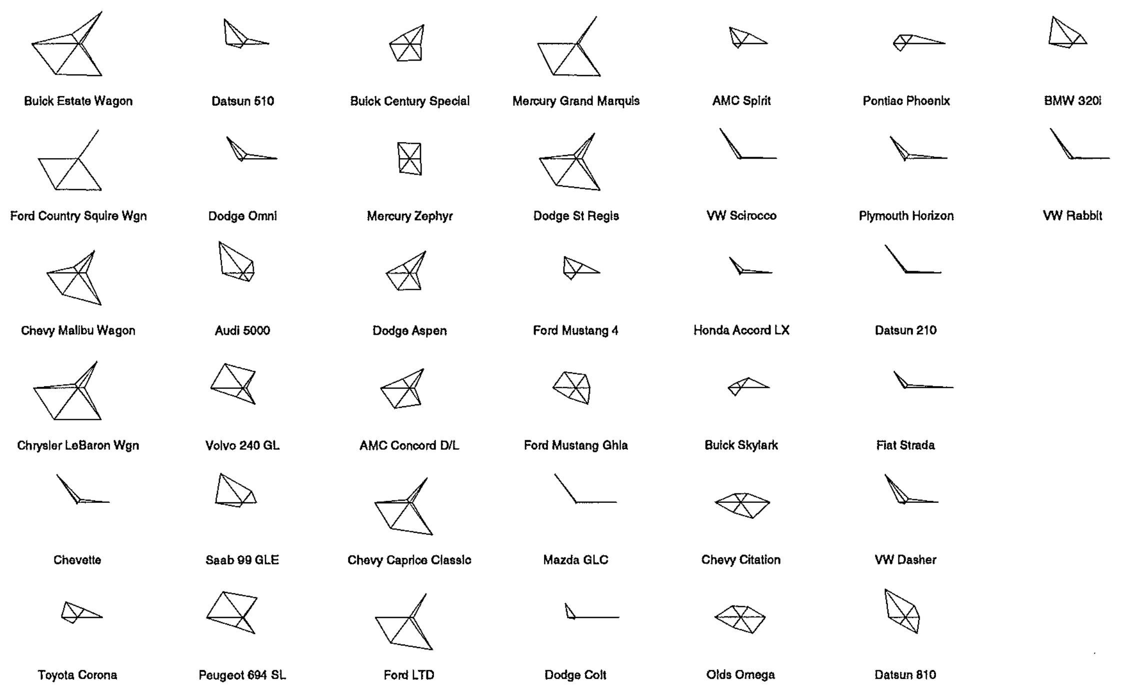
Definite correlations between attributes of the *cars* data table can be seen from Figure 2.4. For example, the weight of a car is proportional to its horsepower, engine displacement, and the number of cylinders, and inversely proportional to its drive ratio and mileage per gallon. Thus, the decision to invert the parallel coordinates for the last two attributes was justified in Section 2.2. The number of cylinders attribute stands out as having only four levels, and separating most other attributes into distinct clusters.

2.6 Iconographic Displays

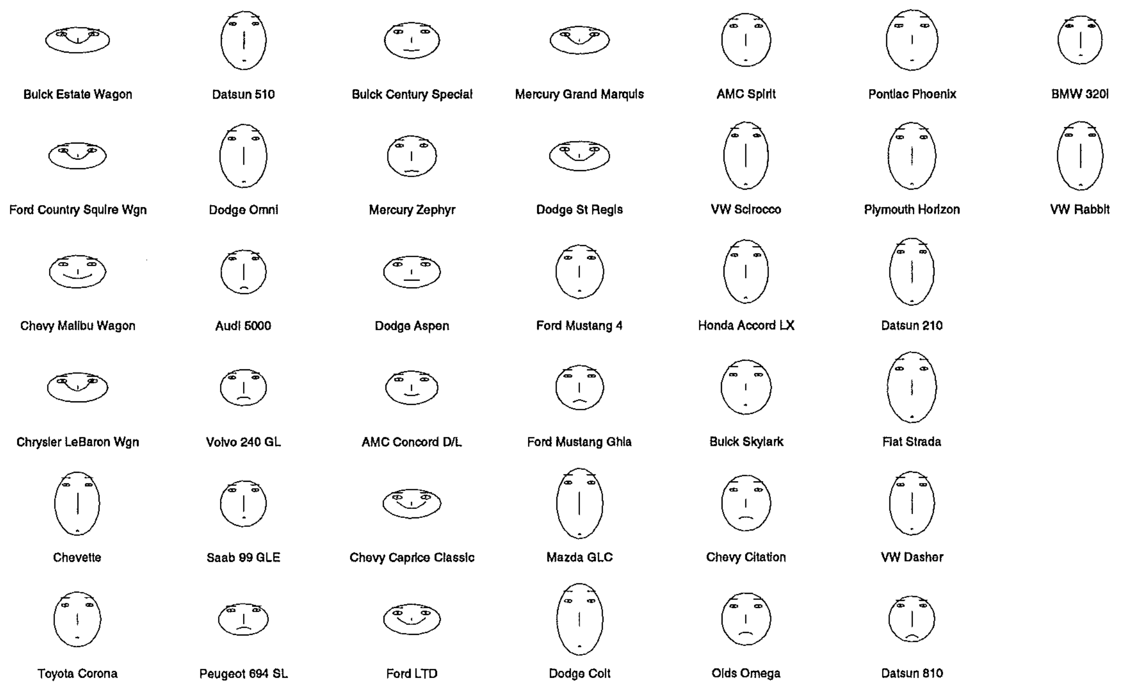
In an iconographic display each icon or glyph represents a single row of a data table. Icons can be arranged in a grid, as in Figures 2.5(a) and 2.5(b), to enable a systematic assessment of similarities and differences between the rows, and also between the attributes. Alternatively, the position of glyphs in the plane can be driven by two of the attributes, providing their spatial interpretation is meaningful. An iconographic display can be combined with the corresponding proximity visualisation, by using icons instead of labelled points, to give the resulting visual representation a degree of redundancy.

2.6.1 Star Glyphs

A star is composed of equally spaced radii, as many as the number of attributes in the data table, stemming from the centre. The length of the rightmost spike is proportional to the value of the first attribute for a given row; the remaining attributes are assigned to their spikes counter clockwise in this manner [Fienberg79].



(a) star glyphs



(b) Chernoff faces

Figure 2.5: Iconographic visualisations

The result of applying this prescription to the *cars* data table is shown in Figure 2.5(a).

The clarity of a star display will suffer as the number of attributes increases, and grouping correlated attributes to provide smooth transitions between spikes might be beneficial. The similarity or dissimilarity of a pair of stars can be appreciated visually; however, gaining a proper overview of a large data table can become a tedious task. This sort of processing is best left to the computer, so that proximity between rows of a data table can be represented in a direct spatial form, as in Section 2.4.

Stars of Figure 2.5(a) can be classified into a few tight groups. This clustering would become more obvious with the aid of automatic or interactive sorting of stars, to bring the similar ones together. However, this will amount to carrying out multidimensional scaling, as pointed out earlier. An interesting observation is that roughly circular stars, e.g. the one for 'Ford Mustang Ghia', appear in the middle of the proximity visualisation of Figure 2.3; many more analogies can be found in both visualisations.

2.6.2 Chernoff Faces

Chernoff faces take advantage of the natural familiarity and recognition of human faces [Chernoff73]. Each facial feature represents one variable; obviously, some features are more prominent, and a possible assignment in the decreasing order of importance is:

- area of the face
- shape of the face
- length of the nose
- location of the mouth
- curve of the smile
- width of the mouth
- location, separation, angle, shape, and width of the eyes
- location of the pupil
- location, angle, and width of the eyebrows

In total, 15 attributes can be represented, and additional variables could be encoded by making faces asymmetric [Flury81]. The trouble is that the appearance of a face will vary with the order of assignment of variables to facial expressions, and perceived similarity of faces will be affected.

Figure 2.5(b) is a collection of Chernoff faces representing the rows of the *cars* data table. Only the first six facial features are used, and the rest is set to a neutral expression. Overall, faces are as effective at portraying similarities as star glyphs. The differences between rows can be detected; however, their magnitude is much more difficult to judge, without detailed knowledge of the assignment of attributes to facial features. Also, it is not possible to tell anymore which icons represent average or extreme observations, e.g. compare 'Ford Mustang Ghia' and 'VW Rabbit'. Additionally, Chernoff faces share disadvantages of star glyphs, and thus are inferior.

2.7 Summary

The advantage of multidimensional scaling over other multivariate visualisation techniques is that it is independent of the number of variables. As long as it is possible to ascertain the high dimensional distance between observations, by using dissimilarity coefficients of Section 1.2 for example, a low dimensional embedding can be found. The type of variables is also immaterial, and even heterogeneous data can be visualised with the aid of the general dissimilarity coefficient (1.8), including nominal variables, which elude other multivariate visualisation methods.

The multidimensional scaling technique scales well with the number of observations, since labelled points or small icons constitute the visual representation of individual observations. Thus, identification of observations is provided, and their actual relationships are represented by proximity. With more observations, the density of icons will increase; however, their relative proximity will be unaffected. Therefore, an informative overview of the data set is presented, highlighting clusters and outliers. Interesting groups of observations can then be analysed separately with this or other multivariate visualisation techniques.

Chapter 3

Multidimensional Scaling

This chapter presents an evaluation of algorithms for generating proximity visualisations. Dissimilarities between pairs of objects from a data collection are given by a suitable coefficient, and then approximated by distances between corresponding pairs of entities in a visual representation. The quality of this approximation is expressed as a loss function, which yields the optimal arrangement at its minimum. The algorithms under test are heuristics for numerically minimising this function. Their effectiveness is measured for a representative sample of data collections, and the relative ranking is statistically inferred, as well as illustrated with examples.

3.1 Problem Definition

Suppose that we are given a collection of n objects and a way of determining the dissimilarity between any pair $\boldsymbol{\Delta} = [\delta_{ij} : i, j = 1, \ldots, n]$; as set out in Section 1.2. Metric Multidimensional Scaling (MDS) [Borg97, Cox94] is a procedure for finding a configuration $\boldsymbol{X}^* = [x^*_{ia} : a = 1, \ldots, p]$ of n points in a p-dimensional space, usually Euclidean, such that point $\boldsymbol{x}^*_i = (x^*_{i1}, \ldots, x^*_{ip})^T$ uniquely represents object i, and the Euclidean distance between points $\boldsymbol{x}^*_i$ and $\boldsymbol{x}^*_j$:

$$d_{ij}(\boldsymbol{X}^*) = \left\| \boldsymbol{x}^*_i - \boldsymbol{x}^*_j \right\| = \sqrt{\sum_{a=1}^{p} \left(x^*_{ia} - x^*_{ja} \right)^2} \tag{3.1}$$

approximates the corresponding dissimilarity δ_{ij}, for all pairs of objects (i, j):

$$\underset{i<j}{\forall}\, d_{ij}(\boldsymbol{X}^*) \approx \delta_{ij} \tag{3.2}$$

In general it is sufficient to consider each pair of objects (i, j) just once: $i < j$, since dissimilarities are assumed to be symmetric. Elements of an asymmetric matrix $\hat{\boldsymbol{\Delta}}$ have to be averaged out prior to the analysis: $\delta_{ij} = \delta_{ji} = (\hat{\delta}_{ij} + \hat{\delta}_{ji})/2$ [Borg97].

3.2 Loss Functions

For a given configuration $\boldsymbol{X}$ the approximation error in representing the dissimilarity between objects i and j can be defined as follows:

$$e_{ij} \overset{\text{def}}{=} |d_{ij}(\boldsymbol{X}) - \delta_{ij}| \tag{3.3}$$

A *least-squares MDS* technique defines a loss function that is a weighted and possibly normalised sum of errors over all pairs of objects (i, j), and thus it penalises the overall approximation error. A minimum of this function over $\boldsymbol{X}$ is subsequently found through numerical optimisation, to yield the desired configuration $\boldsymbol{X}^*$.

3.2.1 Raw Stress

Raw Stress [Kruskal64] is the most elementary MDS loss function, as it simply accumulates the total squared representation error:

$$\sigma_r(\boldsymbol{X}) \overset{\text{def}}{=} \sum_{i<j} e_{ij}^2 = \sum_{i<j} \left(d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}\right)^2 = \sum_{i<j} \left(d_{ij}(\boldsymbol{X}) - f(\delta_{ij})\right)^2 \tag{3.4}$$

where $\hat{d}_{ij}$ is the *disparity* between objects i and j, which is arrived at by applying transformation f to the given dissimilarity δ_{ij}. Since dissimilarities are calculated from object attributes, relationships, or other features (see Section 1.2), and there is no error or uncertainty associated with this process, the correct way to model the disparities is to apply a linear transformation to the dissimilarities: $\hat{d}_{ij} = f(\delta_{ij}) = a\delta_{ij}$, where a is chosen to minimise the value of Stress.

The optimal a, which shall be denoted as a^*, can be found analytically by differentiation. Alternating this step with an iterative improvement to $\boldsymbol{X}$ provides an efficient procedure for finding the solution $\boldsymbol{X}^*$ satisfying $\boldsymbol{\Delta}$, termed *ratio MDS* [Borg97]. By considering the dissimilarities directly, we restrict ourselves to an identity transformation, and arrive at an *absolute MDS* model. At the other end of the spectrum, the transformation can be relaxed just to be monotonic, so that only the rank order of dissimilarities is preserved, because it is assumed to be the only reliable information (see Section 1.2.9), which gives an *ordinal* (*nonmetric*) *MDS* model. Nonmetric MDS can be approximated with the ratio model by replacing the dissimilarities with their rank order a priori [Weeks79, Borg97]. Therefore, we conclude that ratio MDS is the most applicable for visualisation, and focus our efforts on this model.

3.2.2 Normalised Stress

Raw Stress (3.4) can only be successfully minimised in practice under the absolute MDS model. If we permit a transformation of the dissimilarities then by alternating an update of the configuration $\boldsymbol{X}$ with an update to the disparities $[\hat{d}_{ij}] = a\boldsymbol{\Delta}$

$\sigma_r(\boldsymbol{X}, a)$ can be made arbitrarily small, with a and $\boldsymbol{X}$ shrinking gradually. This deficiency can be overcome by normalising raw Stress by $\sum_{i<j} \hat{d}_{ij}^2$ [Borg97]:

$$\sigma_n(\boldsymbol{X}) \stackrel{\text{def}}{=} \frac{\sum_{i<j}\left(d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}\right)^2}{\sum_{i<j} \hat{d}_{ij}^2} \tag{3.5}$$

Loss function (3.5), termed *normalised Stress*, expands for ratio MDS to:

$$\begin{aligned}
\sigma_n(\boldsymbol{X}, \alpha) &= \frac{\sum_{i<j}\left(d_{ij}(\boldsymbol{X}) - \alpha\delta_{ij}\right)^2}{\sum_{i<j} \alpha^2\delta_{ij}^2} \\
&= \frac{\sum_{i<j} d_{ij}^2(\boldsymbol{X}) - 2\alpha\sum_{i<j}\delta_{ij}d_{ij}(\boldsymbol{X}) + \alpha^2\sum_{i<j}\delta_{ij}^2}{\alpha^2\sum_{i<j}\delta_{ij}^2} \\
&= \frac{\eta^2(\boldsymbol{X}) - 2\alpha\rho(\boldsymbol{X}) + \alpha^2\eta_\delta^2}{\alpha^2\eta_\delta^2} \\
&= 1 - \frac{2\alpha\rho(\boldsymbol{X}) - \eta^2(\boldsymbol{X})}{\alpha^2\eta_\delta^2}
\end{aligned} \tag{3.6}$$

where $\eta^2(\boldsymbol{X})$ is a sum of the squared distances $d_{ij}^2(\boldsymbol{X})$, $\rho(\boldsymbol{X})$ is a weighted sum of distances $\delta_{ij}d_{ij}(\boldsymbol{X})$, and η_δ^2 is a constant equal to the sum of the squared dissimilarities δ_{ij}^2, over all pairs of objects (i,j). The optimal α can be found by differentiating (3.6) with respect to α:

$$\begin{aligned}
\frac{\partial\sigma_n(\boldsymbol{X}, \alpha)}{\partial\alpha} &= \frac{-2\alpha^2\rho(\boldsymbol{X}) - 2\alpha\left(-2\alpha\rho(\boldsymbol{X}) + \eta^2(\boldsymbol{X})\right)}{\alpha^4\eta_\delta^2} \\
&= 2\frac{\alpha\rho(\boldsymbol{X}) - \eta^2(\boldsymbol{X})}{\alpha^3\eta_\delta^2}
\end{aligned}$$

which is equal to zero for $\alpha^* = \eta^2(\boldsymbol{X})/\rho(\boldsymbol{X})$. Inserting α^* into (3.6) yields:

$$\begin{aligned}
\sigma_n(\boldsymbol{X}, \alpha^*) &= 1 - \frac{2\rho(\boldsymbol{X})\dfrac{\eta^2(\boldsymbol{X})}{\rho(\boldsymbol{X})} - \eta^2(\boldsymbol{X})}{\eta_\delta^2\dfrac{\eta^4(\boldsymbol{X})}{\rho^2(\boldsymbol{X})}} \\
&= 1 - \left(\frac{\rho(\boldsymbol{X})}{\eta_\delta\eta(\boldsymbol{X})}\right)^2 \\
&= 1 - \left(\frac{\sum_{i<j}\delta_{ij}d_{ij}(\boldsymbol{X})}{\left(\sum_{i<j}\delta_{ij}^2\right)^{\frac{1}{2}}\left(\sum_{i<j}d_{ij}^2(\boldsymbol{X})\right)^{\frac{1}{2}}}\right)^2 \\
&= 1 - c\left(\boldsymbol{\Delta}, [d_{ij}(\boldsymbol{X})]\right)^2
\end{aligned} \tag{3.7}$$

The last term of (3.7) is the square of Tucker's congruence coefficient c with dissimilarities and distances [Tucker51]. The coefficient is always between 0 and

1, due to the Cauchy-Schwarz inequality:

$$\sum_r p_r q_r \leq \left(\sum_r p_r^2\right)^{\frac{1}{2}} \left(\sum_r q_r^2\right)^{\frac{1}{2}} \tag{3.8}$$

and the fact that dissimilarities and distances are nonnegative. Hence, it holds that $0 \leq \sigma_n(\boldsymbol{X}, \alpha^*) \leq 1$, for the optimal scaling constant α.

It is worth noting that scaling the dissimilarities by a factor α is equivalent to scaling the distances by $1/\alpha$:

$$\begin{aligned}
\sigma_n(\boldsymbol{X}, \alpha) &= \frac{\sum_{i<j} (d_{ij}(\boldsymbol{X}) - \alpha\delta_{ij})^2}{\sum_{i<j} \alpha^2 \delta_{ij}^2} \\
&= \frac{\sum_{i<j} (\alpha^{-1} d_{ij}(\boldsymbol{X}) - \delta_{ij})^2}{\sum_{i<j} \delta_{ij}^2} \\
&= \frac{\sum_{i<j} (d_{ij}(\alpha^{-1}\boldsymbol{X}) - \delta_{ij})^2}{\sum_{i<j} \delta_{ij}^2} \quad = \quad \sigma_n(\alpha^{-1}\boldsymbol{X})
\end{aligned}$$

The derivation of $\sigma_n(b^*\boldsymbol{X})$ is equivalent to that presented above [deLeeuw77, Borg97], and yields $b^* = 1/\alpha^*$. However, it relies on the assumption that $\boldsymbol{X}$ is a local minimum of (3.5), whereas our derivation does not require such an assumption, and thus is applicable to any configuration $\boldsymbol{X}$.

3.2.3 Kruskal's Stress

Stress formula 1, or *Stress-1* for short, is historically an earlier solution to the scale dependency problem of raw Stress (3.4), with the normalising factor $\eta^2(\boldsymbol{X})$ [Kruskal64]:

$$\sigma_1^2(\boldsymbol{X}) \overset{\text{def}}{=} \frac{\sum_{i<j} \left(d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}\right)^2}{\sum_{i<j} d_{ij}^2(\boldsymbol{X})} \tag{3.9}$$

The whole formula is square rooted by analogy of referring to the standard deviation instead of the variance.

Substituting the ratio MDS model yields:

$$\begin{aligned}
\sigma_1^2(\boldsymbol{X}, \beta) &= \frac{\sum_{i<j} (d_{ij}(\boldsymbol{X}) - \beta\delta_{ij})^2}{\sum_{i<j} d_{ij}^2} \\
&= \frac{\eta^2(\boldsymbol{X}) - 2\beta\rho(\boldsymbol{X}) + \beta^2\eta_\delta^2}{\eta^2(\boldsymbol{X})} \\
&= 1 - \frac{2\beta\rho(\boldsymbol{X}) - \beta^2\eta_\delta^2}{\eta^2(\boldsymbol{X})}
\end{aligned} \tag{3.10}$$

The minimum of (3.10) over β is obtained by setting the first derivative with respect to β equal to 0:

$$\frac{\partial\sigma_1^2(\boldsymbol{X}, \beta)}{\partial\beta} = 2\frac{\beta\eta_\delta^2 - \rho(\boldsymbol{X})}{\eta^2(\boldsymbol{X})} = 0$$

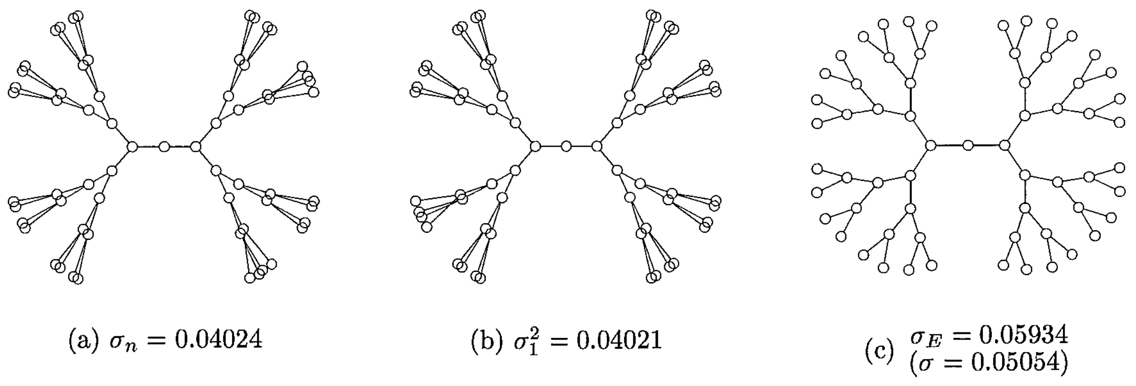

(a) $\sigma_n = 0.04024$
(b) $\sigma_1^2 = 0.04021$
(c) $\sigma_E = 0.05934$ $(\sigma = 0.05054)$

Figure 3.1: Visualisations of a 6 level complete binary tree computed by minimising different loss functions. The dissimilarity between a pair of nodes is the length of the path connecting them (see Section 1.2.6)

which yields $\beta^* = \rho(\boldsymbol{X})/\eta_\delta^2$. Inserting β^* into (3.10) gives:

$$\sigma_1^2(\boldsymbol{X}, \beta^*) = 1 - \left(\frac{\rho(\boldsymbol{X})}{\eta(\boldsymbol{X})\eta_\delta}\right)^2 = 1 - c\left(\Delta, [d_{ij}(\boldsymbol{X})]\right)^2 = \sigma_n(\boldsymbol{X}, \alpha^*) \stackrel{\text{def}}{=} \sigma \tag{3.11}$$

Therefore, for a given configuration $\boldsymbol{X}$, normalised Stress (3.5) is equivalent to the square of Stress-1 (3.9) if we allow for an optimal scaling of dissimilarities in each case, and we shall refer to this unified index as Stress (σ). Since such a scaling step is an integral part of any ratio MDS procedure, and is alternated with an iterative improvement to $\boldsymbol{X}$, the final solution will be a local minimum of both of these loss functions. A graphical illustration of this equivalence is presented in Figure 3.1(a) and 3.1(b). The relationship (3.11) can also be demonstrated by taking the alternative route of rescaling the configuration $\boldsymbol{X}$ [Borg97], however an assumption that $\boldsymbol{X}$ is a local minimum has to be made again.

3.2.4 Energy

Any of the loss functions defined previously can be generalised to weight the contribution of individual pairs of objects, for example weighted normalised Stress acquires the following form:

$$\sigma_{n,w}(\boldsymbol{X}) \stackrel{\text{def}}{=} \frac{\sum_{i<j} w_{ij}\left(d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}\right)^2}{\sum_{i<j} w_{ij}\hat{d}_{ij}^2} \tag{3.12}$$

Each weight w_{ij} can take any non-negative value, as long as it does not depend on $\boldsymbol{X}$. Typical use allows missing dissimilarities to be accommodated by setting the corresponding weights to 0, and the remaining ones to 1. However, by setting

$w_{ij} = \hat{d}_{ij}^{-2}$ a new loss function can be obtained, which we shall refer to as *Energy*:

$$\sigma_E(\boldsymbol{X}) \overset{\text{def}}{=} \frac{2}{n(n-1)} \sum_{i<j} \frac{\left(d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}\right)^2}{\hat{d}_{ij}^2} \tag{3.13}$$

The ratio MDS model produces the following expansion, with $\frac{1}{\lambda} = \frac{n(n-1)}{2} = \binom{n}{2} = \sum_{i<j} 1$:

$$\begin{aligned}
\sigma_E(\boldsymbol{X}, \gamma) &= \lambda \sum_{i<j} \frac{\left(d_{ij}(\boldsymbol{X}) - \gamma\delta_{ij}\right)^2}{\gamma^2 \delta_{ij}^2} \\
&= \lambda \left(\sum_{i<j} 1 - \sum_{i<j} \frac{2\gamma\delta_{ij} d_{ij}(\boldsymbol{X}) - d_{ij}^2(\boldsymbol{X})}{\gamma^2\delta_{ij}^2} \right) \\
&= 1 - \lambda \left(\frac{2}{\gamma} \sum_{i<j} \frac{d_{ij}(\boldsymbol{X})}{\delta_{ij}} - \frac{1}{\gamma^2} \sum_{i<j} \frac{d_{ij}^2(\boldsymbol{X})}{\delta_{ij}^2} \right) \\
&= 1 - \lambda \left(\frac{2}{\gamma} \psi(\boldsymbol{X}) - \frac{1}{\gamma^2} \omega(\boldsymbol{X}) \right)
\end{aligned} \tag{3.14}$$

An optimal γ can be found by differentiating (3.14) with respect to γ:

$$\begin{aligned}
\frac{\partial \sigma_E(\boldsymbol{X}, \gamma)}{\partial \gamma} &= -\lambda \left(-\frac{2}{\gamma^2} \psi(\boldsymbol{X}) + \frac{2}{\gamma^3} \omega(\boldsymbol{X}) \right) \\
&= \frac{2\lambda}{\gamma^3} \left(\gamma\psi(\boldsymbol{X}) - \omega(\boldsymbol{X}) \right)
\end{aligned}$$

which is equal to zero for $\gamma^* = \omega(\boldsymbol{X})/\psi(\boldsymbol{X})$. Substituting γ^* into (3.14) yields:

$$\begin{aligned}
\sigma_E(\boldsymbol{X}, \gamma^*) &= 1 - \lambda \frac{\psi^2(\boldsymbol{X})}{\omega(\boldsymbol{X})} \\
&= 1 - \lambda \frac{\left(\sum_{i<j} \frac{d_{ij}(\boldsymbol{X})}{\delta_{ij}} \right)^2}{\sum_{i<j} \frac{d_{ij}^2(\boldsymbol{X})}{\delta_{ij}^2}} \\
&= 1 - \frac{\left(\sum_{i<j} \frac{\delta_{ij} d_{ij}(\boldsymbol{X})}{\delta_{ij}^2} \right)^2}{\sum_{i<j} \frac{\delta_{ij}^2}{\delta_{ij}^2} \sum_{i<j} \frac{d_{ij}^2(\boldsymbol{X})}{\delta_{ij}^2}} \\
&= 1 - \left(\frac{\sum_{i<j} w_{ij}\delta_{ij} d_{ij}(\boldsymbol{X})}{\left(\sum_{i<j} w_{ij}\delta_{ij}^2\right)^{\frac{1}{2}} \left(\sum_{i<j} w_{ij} d_{ij}^2(\boldsymbol{X})\right)^{\frac{1}{2}}} \right)^2 \\
&= 1 - c_w\left(\boldsymbol{\Delta}, [d_{ij}(\boldsymbol{X})]\right)^2
\end{aligned} \tag{3.15}$$

The last term of (3.15) is the square of weighted Tucker's congruence coefficient c_w with dissimilarities and distances [Tucker51]. The coefficient is always between 0 and 1, due to the Cauchy-Schwarz inequality (3.8). Hence, it holds that $0 \leq \sigma_E(\boldsymbol{X}, \gamma^*) \leq 1$, for the optimal scaling constant γ.

Energy (3.13) penalises error (3.3) in representing short dissimilarities more than the same error for larger dissimilarities. Such a proportional contribution of error is likely to agree with user's expectations, because it does not matter exactly how distant dissimilar objects are from a given object, as long as they are far in relation to similar ones. Figure 3.1 gives a visual comparison of the results of minimising Stress (3.11) and Energy. Stress, on the other hand, minimises absolute error, since errors in representing large and small dissimilarities are penalised equally, and does not preserve the local detail as well as Energy. Therefore, we prefer to use Energy for proximity visualisation, with the exception of Chapter 5.

For simplicity we drop the normalising constant λ when deriving MDS algorithms, and minimise raw Energy instead:

$$\sigma_{Er}(\boldsymbol{X}) \stackrel{\text{def}}{=} \sum_{i<j} \frac{\left(d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}\right)^2}{\hat{d}_{ij}^2} \tag{3.16}$$

which can be derived from the weighted form of raw Stress (3.4):

$$\sigma_{r,w}(\boldsymbol{X}) \stackrel{\text{def}}{=} \sum_{i<j} w_{ij} \left(d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}\right)^2 \tag{3.17}$$

by setting the appropriate weights. We note that minimising (3.16) is equivalent to minimising (3.13), and we use the normalised form when reporting results.

(3.16) may be interpreted as the total energy of a fully connected spring system, with an anchor for each object, and springs connecting it to all other anchors. The relaxed length of a spring connecting two anchors is given by the optimally scaled dissimilarity between the corresponding pair of objects. The actual length of the spring is the Euclidean distance between the anchors. The spring constant is $\hat{d}_{ij}^{-2}$ for a spring connecting anchors i and j, which causes the spring to be rigid if δ_{ij} is small in relation to other dissimilarities, and therefore to contribute substantially to the value of (3.16) if deformed. Equilibrium of this spring system corresponds to a local energy minimum.

3.3 Algorithms

Each of the least-squares MDS algorithms presented here is an instance of a well-known function minimisation heuristic. Such a heuristic constitutes a theoretical framework, and typically a large number of options or variations contributed by practitioners in the operational research and other fields. Some of the choices are simply determined by the application domain, others can be made on theoretical grounds, or empirical evidence – either found in literature or established on our own. In each case we outline the heuristic framework, giving references to dedicated texts, and detail the decisions and customisations made to provide a complete algorithm.

3.3.1 Classical Scaling

Classical scaling [Gower66, Cox94, Borg97] is an analytical technique for solving the metric MDS problem, and it was the first to be developed. Dissimilarities are treated as Euclidean distances, and a matching configuration $\boldsymbol{Z}$ is recovered from the eigendecomposition of $-\frac{1}{2}\boldsymbol{J}\boldsymbol{\Delta}^{(2)}\boldsymbol{J} = \boldsymbol{Z}\boldsymbol{Z}^T$, where $\boldsymbol{J}$ is the centring matrix, and $\boldsymbol{\Delta}^{(2)} = [\delta_{ij}^2]$. $p \leq n-1$ eigenvalues will be non-zero, and they represent the minimum number of dimensions required to preserve all the dissimilarities, so that (3.2) holds exactly. Selecting $p' < p$ largest eigenvalues and corresponding eigenvectors is equivalent to selecting the first p' principal components $\boldsymbol{Y}$ of the p-dimensional configuration $\boldsymbol{Z}$, i.e. the projection of $\boldsymbol{Z}$ onto p' orthogonal axes that preserves the maximum variance (see Section 4.3). This combined procedure is termed Principal Coordinate Analysis (PCO)[†] [Gower66], and analytically minimises the following criterion over $\boldsymbol{Y}$:

$$\sum_{i<j}\left(d_{ij}^2(\boldsymbol{Z}) - d_{ij}^2(\boldsymbol{Y})\right) \equiv \sum_{i<j}\left(\delta_{ij}^2 - d_{ij}^2(\boldsymbol{Y})\right) \tag{3.18}$$

Note, that under the projection $d_{ij}(\boldsymbol{Y}) \leq d_{ij}(\boldsymbol{Z}) = \delta_{ij}$, for all pairs (i,j), i.e. distortion of the dissimilarities will be biased, unlike for the least-squares MDS loss functions of Section 3.2.

3.3.2 Newton-Raphson

The initial MDS algorithm that we have developed is based on the generalised Newton-Raphson (NR) method [Press92]. It is an iterative technique for solving a vector equation of the form $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. If $\boldsymbol{f}(\boldsymbol{x})$ represents the first partial derivatives of a multidimensional function $e(\boldsymbol{x})$ this procedure will find an extremum of this function.

The gradient vector $\boldsymbol{g}$ and the Hessian matrix $\boldsymbol{H}$ of loss function (3.16) at point $\boldsymbol{x}_k = (x_{k1}, \ldots, x_{kp})^T$ can be established analytically (see Appendix A for details):

$$\begin{aligned} g_a(\boldsymbol{x}_k) &= \frac{\partial \sigma_{Er}(\boldsymbol{X})}{\partial x_{ka}} = 2\sum_{l\neq k}\frac{d_{kl}(\boldsymbol{X}) - \hat{d}_{kl}}{d_{kl}(\boldsymbol{X})\hat{d}_{kl}^2}(x_{ka} - x_{la}) \\ H_{ab}(\boldsymbol{x}_k) &= \frac{\partial^2 \sigma_{Er}(\boldsymbol{X})}{\partial x_{ka}\partial x_{kb}} = \begin{cases} 2\sum_{l\neq k}\frac{(x_{ka}-x_{la})(x_{kb}-x_{lb})}{d_{kl}^3(\boldsymbol{X})\hat{d}_{kl}} & : \ a \neq b \\ 2\sum_{l\neq k}\frac{1}{\hat{d}_{kl}^2} - 2\sum_{l\neq k}\frac{d_{kl}^2(\boldsymbol{X}) - (x_{ka}-x_{la})^2}{d_{kl}^3(\boldsymbol{X})\hat{d}_{kl}} & : \ a = b \end{cases} \end{aligned}$$

[†]in our work we have used an implementation of PCO – DistPCoA – made publicly available by Philippe Casgrain: www.fas.umontreal.ca/BIOL/Casgrain/en/labo/distpcoa.html

An extremum of the loss function with respect to point $\boldsymbol{x}_k$ can then found by repeatedly applying the NR iteration:

$$\begin{aligned} \tilde{\boldsymbol{x}}_k &= \boldsymbol{x}_k - \boldsymbol{H}^{-1}\boldsymbol{g} \\ \boldsymbol{H}\left(\tilde{\boldsymbol{x}}_k - \boldsymbol{x}_k\right) &= -\boldsymbol{g} \end{aligned}$$

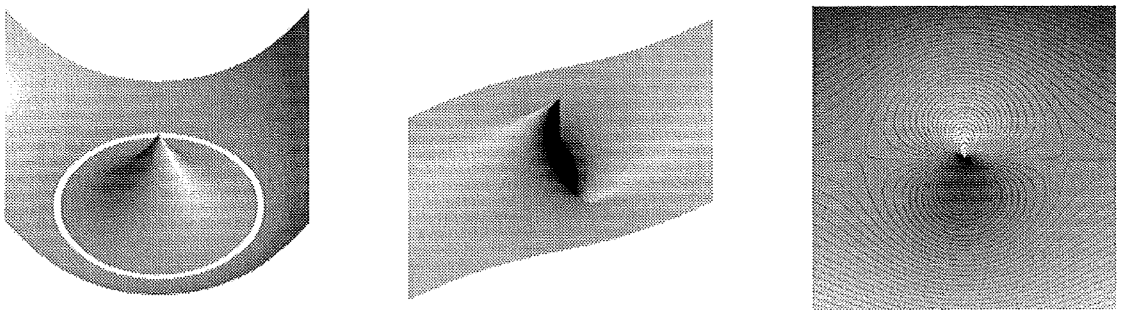
Let the i^{th} column of $\boldsymbol{H}$ be denoted by $\boldsymbol{H}_i = (H_{1i}, \ldots, H_{pi})^T$

$$\begin{aligned} \tilde{\boldsymbol{x}}_k &= \boldsymbol{x}_k + \left(\frac{\det(-\boldsymbol{g}, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_p)}{\det \boldsymbol{H}}, \frac{\det(\boldsymbol{H}_1, -\boldsymbol{g}, \ldots, \boldsymbol{H}_p)}{\det \boldsymbol{H}}, \right. \\ &\quad \left. \ldots, \frac{\det(\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, -\boldsymbol{g})}{\det \boldsymbol{H}} \right)^T \end{aligned} \tag{3.19}$$

Figure 3.2(a) depicts the Energy function (3.13) for a trivial case when there is a single constant (fixed) point $\boldsymbol{x}_f$ to optimise $\boldsymbol{x}_k$ against in 2 dimensions. A circle of radius $\hat{d}_{fk}$ centred at $\boldsymbol{x}_f$ is the set of all minima. There is a local maximum at $\boldsymbol{x}_f$ where the function takes value 1. However, it is a singular point, as is apparent from Figures 3.2(b) and 3.2(c); consequently, gradient $\boldsymbol{g}$ does not have a root at $\boldsymbol{x}_f$. Moreover, if $\boldsymbol{x}_k$ gets inside the minimum circle it will be repelled with a subsequent NR iteration (3.19), because the direction of $\boldsymbol{g}$ is reversed over the cusp at point $\boldsymbol{x}_f$, whereas Hessian $\boldsymbol{H}$ is not affected. This argument extends to the case of multiple constant points; see Figure 3.3 for an illustration. Because of the parabolic characteristic of the component Energy functions, no new maxima can be introduced by their summation, and thus the NR method will only converge to a minimum. Unlike the steepest descent method [Press92] (also see Section 3.3.5) an upward move – a transition from $\boldsymbol{x}_k$ to $\tilde{\boldsymbol{x}}_k$ that increases Energy – is possible with an NR iteration, especially if the current solution is far from a minimum. This will prevent the method from getting stuck in a poor local minimum, in general.

To minimise the loss function with respect to the whole configuration, the algorithm applies a single NR iteration (3.19) to each point in turn, and continually cycles through the configuration until convergence occurs. Since the focus of our work is visualisation, we only need to consider MDS in at most three dimensions. By deriving analytical formulae for up to 3 variables and hardcoding them, we have been able to substantially increase the overall efficiency of the algorithm. An algorithm for drawing general undirected graphs [Kamada89] also minimises (3.16) using the NR method, and its basic details are identical.

The sequence in which points are considered when updating the configuration is randomised to ensure faster convergence [Frick94]. This has a side effect of randomising output of the algorithm, even if the same starting configuration is used. To detect convergence a record of previous configurations is kept. The algorithm terminates if the current configuration is identical to a prior one, to within a certain level of precision ϵ (*NR.epsilon* parameter); this can be triggered either by a genuine convergence or an occurrence of a cycle. Using a hash value of a complete configuration minimises the storage requirements of this scheme,



(a) Surface plot of $\sigma_E(\boldsymbol{x}_k)$ (b) Surface plot of $\partial\sigma_E(\boldsymbol{x}_k)/\partial x_{k1}$ (c) Contour plot of $\partial\sigma_E(\boldsymbol{x}_k)/\partial x_{k2}$ (symmetrical to x_{k1} derivative)

Figure 3.2: Plots of the Energy function and its first partial derivatives for one fixed point in 2 dimensions

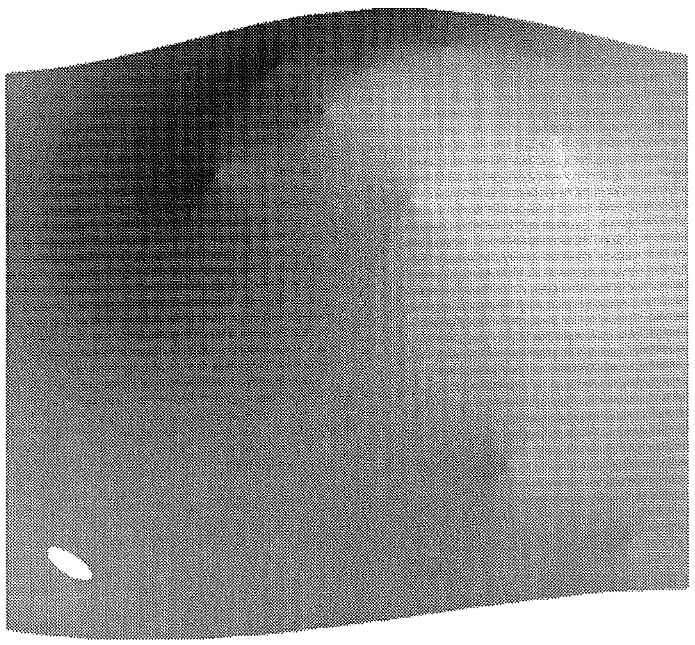


Figure 3.3: Surface plot of $\sigma_E(\boldsymbol{x}_k)$ for 10 fixed points in 2 dimensions. These points can be identified as distinct peaks – local maxima. The white elliptical area encompasses the global minimum

and makes the test for equality of two configurations take constant time. The Secure Hash Algorithm [NIST95] has been chosen for its property of making it computationally infeasible to find two different messages – MDS configurations in this case – that produce the same hash value.

It is possible for an NR iteration (3.19) to result in a radically different new location of a point from its original one, normally causing a drastic increase in Energy (3.13). This will have a knock on effect on points considered subsequently, and may result in a perpetual instability of the configuration. This would not constitute a cycle in general, and therefore to trap this behaviour a separate mechanism is required. A cyclic buffer of size l (*NR.buffersize* parameter) is used to record the Energy of the l most recent configurations. If the current configuration is found to be inferior or equivalent to the oldest configuration in the buffer, a flag is set. l configuration updates are then performed unconditionally; if the situation does not reoccur within another l updates the flag is cleared. Otherwise, the flag is kept set, and ϵ is increased by one order of magnitude. If instability persists ϵ will increase to a point where any two configurations will be judged equivalent, and the algorithm will necessarily stop.

3.3.3 Tabu Search

Tabu search (TS) [Glover95] is a meta-heuristic that improves on known numerical optimisation heuristics by exploiting principles of intelligent problem solving. The key element is the use of flexible memory to guide the optimisation process around difficult regions, and allow boundaries of local optimality to be crossed, so that new regions of solution space can be explored. This is achieved by systematically imposing and releasing restrictions on certain search moves (transitions from one solution to the next), based on their recency, frequency, quality, and influence. These restrictions can be implemented by directly excluding such moves and classing them as 'tabu' or 'forbidden'; an alternative is to modify loss function evaluations for these moves or the probabilities of their selection.

To illustrate the basic principles of tabu search we will describe the main features of our initial application of this method to MDS. A search move consists of applying a single Newton-Raphson iteration (3.19) to a point in the configuration. Instead of simply cycling through the moves, as is the case for the *NR* algorithm (see Section 3.3.2), they are selected based on the optimisation history. If a move has been recently performed it is classified as tabu, and excluded from consideration for the number of iterations equal to the duration of the tabu restriction. However, an important exception is taken into account: the restriction is revoked if the move would result in the best solution found so far. Such a mechanism for overriding the tabu classification is termed an aspiration criterion. Another aspiration criterion is used alongside the best solution criterion: aspiration by search direction permits a tabu move that improves on the current solution if it also caused improvement the last time it was performed.

For each move its associated value can be determined as the change to the value of raw Energy (3.16) it would cause if it was applied. Of all admissible (non-tabu) moves the one with the highest value is selected, as long as it is positive. If no improving move exists frequency-based memory is used to determine the winner. The moves are sorted in the descending order of their values, and the one with the smallest sum of its rank and frequency (count of its prior occurrence) is selected. A tie is broken by selecting the least frequent move. Frequency memory operates over the long term, and its application here diversifies the search, driving it into new regions.

We found however that this algorithm had poor convergence performance. The cause of the problem was that moves which consistently had large negative values would only be selected after frequency memory disqualified other moves, i.e. after a large number of iterations. Such moves are temporarily disadvantageous but crucial if a minimum is to be found. Consequently, many iterations were being wasted considering only a subset of the configuration. The second shortcoming was that in order to calculate move values incrementally between iterations, for efficiency reasons, a different loss function to raw Energy (3.16) had to be used. This is due to the fact that (3.16) involves a square root calculation, and its update for each move cannot be determined analytically, i.e. in fixed time. We overcame this problem by squaring both terms in the numerator of (3.16):

$$\sigma_{Er^2}(\boldsymbol{X}) \overset{\text{def}}{=} \sum_{i<j} \frac{\left(d_{ij}^2(\boldsymbol{X}) - \hat{d}_{ij}^2\right)^2}{\hat{d}_{ij}^2} \tag{3.20}$$

which can be derived from weighted S-Stress [Takane77] by setting $w_{ij} = \hat{d}_{ij}^{-2}$:

$$\sigma_{r^2}(\boldsymbol{X}) \overset{\text{def}}{=} \sum_{i<j} w_{ij} \left(d_{ij}^2(\boldsymbol{X}) - \hat{d}_{ij}^2\right)^2 \tag{3.21}$$

However, the expanded analytical formulas for updating (3.20) were complicated and expensive to compute.

The reason that we thought TS should perform better than *NR* is that it would not indiscriminately apply the moves, and hence could prevent configuration instability altogether. Since it is important for all points to be updated regularly, but avoid drastic moves (see Section 3.3.2), we decided to rely solely on frequency memory for selection, combined with move values, in the revised version of the algorithm. The value of a move is defined as the contribution of the originating point $\boldsymbol{x}_i$ to the overall value of (3.16), i.e. it is the total energy of springs stemming from it:

$$\sigma_{Er}(\boldsymbol{x}_i) = \sum_{j \neq i} \frac{\left(d_{ij} - \hat{d}_{ij}\right)^2}{\hat{d}_{ij}^2} \tag{3.22}$$

These values can be updated between iterations with little overhead. The move with the lowest sum of its value rank in descending order and its frequency is selected, favouring the least frequent one if there is a tie. This will have an effect of keeping frequency of all moves roughly equal, and making subsequent selection of a drastic move very likely, which will give the corresponding point an opportunity to finally assume a favourable location. The termination criterion is the same as in the *NR* algorithm, i.e. when the configuration reaches stability up to a desired level of numerical precision (*TS.epsilon* parameter).

3.3.4 Genetic Algorithm

Genetic Algorithms (GAs) [Goldberg89, Reeves95] exploit random search in an intelligent manner to solve optimisation problems. Because they work with a coding of the parameter set, and not the parameters themselves, they are generic and largely domain independent. To steer the process only some measure of performance or fitness is needed, for example evaluation of the loss function in the case of a function optimisation problem. GAs draw a parallel to the mechanics of natural evolution. A population of candidate solutions (chromosomes) is maintained, and they are selectively combined by means of a crossover operator into a new generation of solutions. A mutation operator is applied probabilistically to diversify the population, and prevent premature convergence. Although selection is random, it is biased towards solutions with high fitness, giving their parameters (genes) a better chance of propagating to subsequent generations, and therefore influencing the optimisation process. A record of the best solution is kept, and returned upon termination.

Because MDS is a continuous optimisation problem, we have decided to use the *real coding* [Surry95] in our algorithm. A solution is encoded as a vector of floating-point numbers (genes), which directly correspond to coordinates of points in the configuration. *Blend crossover* (BLX) [Surry95] is used for recombining genes. It takes a parameter α that controls its precise effect, and a pair of values x and y to recombine. Assuming $x \leq y$ without loss of generality, its outcome is a uniform random number drawn from the interval $[x - \alpha(y - x), y + \alpha(y - x)]$. BLX_0 operates on the range $[x, y]$, and is therefore biased towards the centre. This effect is countered by using $\alpha = 0.5$. A new solution is derived from a pair of chromosomes by applying $\mathrm{BLX}_{0.5}$ to each pair of corresponding genes.

The most appropriate form of mutation to use with the real coding is *creep mutation* [Surry95]. Instead of replacing the affected gene with a new random value, the current value is perturbed by a small amount. This can be achieved by adding on a number drawn from the Gaussian distribution of zero mean and standard deviation sigma (*GA.sigma* parameter), for example. The crossover operator is most effective when the population is diverse. At this stage of optimisation too high a level of mutation will interfere with crossover. However, once the population starts to converge, crossover becomes ineffective, and a good rate of mutation is essential if different solutions are to be explored, and optimisation to continue [Reeves95]. Therefore, we concluded that an adaptive rate of mutation would work best. Whenever BLX is attempted on a pair of genes that are identical up to a certain level of precision (*GA.epsilon* parameter), Gaussian mutation is performed instead. This form of mutation reintroduces diversity only when needed.

The final facet of the algorithm is the process of selecting which chromosomes will undergo the genetic operations described above, to form the next generation. To steer the optimisation process, it is important to give a higher likelihood of selection to solutions with high fitness $f(\boldsymbol{X}) = c - \sigma_E(\boldsymbol{X})$, where $c \geq \sigma_E(\boldsymbol{X})$ for all $\boldsymbol{X}$. However, simply making the probability of selection proportional to fitness has undesirable effects. Initially, a relatively good solution could take over the population, and cause it to converge prematurely around a poor local minimum. At the later stages of optimisation, when the population has largely converged, and differences between solutions are small, best solutions will not receive enough emphasis. The desired level of competition among members of the population can be attained throughout the algorithm run if rank of solution fitness is used instead, as is effectively done in the *tournament selection* scheme [Reeves95].

Tournament selection takes its name from performing a tournament on a random subset of t solutions from the population of size m (*GA.populationsize* parameter), and choosing the solution with the highest fitness value as the winner. This selection is performed m times, and the subsets are determined by randomly perturbing the population sequence, and taking successive groups of t elements. Once the pool of solutions is exhausted, a new permutation is created; t repetitions are needed overall. Setting $t = 2$ results in an adequate selective pressure [Surry95], in that the best solution will be chosen twice, the worst not at all, and the median once on average.

Successive pairs of tournament winners are subject to the combined crossover and mutation operator twice, to yield two new solutions. m needs to be divisible by $2t$ to ensure that these pairs come from a single permutation, and hence avoid the possibility of a solution being mated with itself. A new generation of m solutions is created in this way, and the algorithm can proceed to the next iteration. If a superior solution is encountered it is noted, otherwise the best solution found so far replaces the worst solution in the new population. This mechanism is a form of elitist selection [deJong75], and guarantees monotonic improvement of maximum population fitness, and ultimately convergence. Termination occurs after a set number of iterations (*GA.iterationlimit* parameter) have been performed without finding an improving solution.

3.3.5 Majorization Algorithm

Iterative Majorization (IM) [Ortega70, deLeeuw77] belongs to a class of descent techniques for function minimisation [Press92], as it generates a non-increasing sequence of function values. If a complicated function f to be minimised is bounded from below, like Energy (3.13) for example, IM will converge to a local minimum of f. Instead of minimising $f(x)$ directly, an auxiliary function $g(x, z)$, where z is a constant, is defined and minimised, to get an approximation to a minimum of $f(x)$. By choosing g to be quadratic in x, its unique minimum can be found analytically, although other function forms can be useful, too. g has to meet two conditions to be called a *majorizing function* of f:

1. $f(x) \leq g(x, z)$, for all x
2. $f(z) = g(z, z)$, with z referred to as the *supporting point*

Assuming that x^* is the minimum of $g(x, z)$ over x, the majorizing conditions imply the following chain of inequalities:

$$f(x^*) \leq g(x^*, z) \leq g(z, z) = f(z) \tag{3.23}$$

Once x^* is found it is used as the new supporting point: $z' = x^*$, and thus a better approximation to the minimum of f is established iteratively.

The principle of IM can be easily generalised to multivariable functions. There exists a quadratic majorizing function for weighted raw Stress (3.17), and it forms the basis of the SMACOF algorithm [deLeeuw77, Borg97] (the acronym stands for Scaling by MAjorizing a COmplicated Function). The scale dependency of (3.17) (see Section 3.2.2) is circumvented by explicitly normalising $\{\hat{d}_{ij}\}$, so that $\sum_{i<j} w_{ij}\hat{d}^2_{ij} = n(n-1)/2$. By setting each $w_{ij} = \delta_{ij}^{-2}$ this algorithm can be made to minimise raw Energy (3.16). The explicit normalisation under the ratio MDS model $(\hat{d}_{ij} = a\delta_{ij})$ becomes:

$$\begin{aligned} \sum_{i<j} \delta_{ij}^{-2} a^2 \delta_{ij}^2 &= \frac{n(n-1)}{2} \\ a^2 \sum_{i<j} 1 &= \sum_{i<j} 1 \\ a &= 1 \end{aligned}$$

that is we arrive at the absolute MDS model with loss function (3.16).

The most recent implementation of the SMACOF algorithm is ProxScal version 6.4 [Busing97], and we decided to use it as a benchmark for our MDS algorithms (see section 3.5). The stopping criteria for the algorithm are met when either the rate of change of Energy (3.13) drops below a specified threshold value (*ProxScal.rate* parameter), or a maximum number of iterations has been reached (*ProxScal.iterationlimit* parameter).

3.3.6 Simulated Annealing

Simulated Annealing (SA) [Dowsland95] generalises descent methods of local optimisation (see Section 3.3.5 for an example) by allowing uphill moves, i.e. ones increasing the value of the loss function. It is hoped that by introducing such moves the method will be able to escape local minima, and ultimately converge to a global minimum[‡]. To satisfy this objective the acceptance probability for an uphill move has to be related to the magnitude of the increase, and the optimisation history. A convenient formula can be borrowed from thermodynamics:

$$p(\delta E) = \exp\left(-\frac{\delta E}{kT}\right) \tag{3.24}$$

[‡]we use the indefinite article because there may be many distinct solutions with the same globally minimal value of the loss function

which gives the probability of an increase δE in energy at temperature T. k is Boltzmann's constant, and is obviously meaningless in the function minimisation context, thus the whole denominator is replaced by a single parameter t, although the thermal nomenclature is retained. For negative δE the formula (3.24) produces $p(\delta E) > 1$, thus a downhill move is always accepted. Initially the temperature is high, and so is the probability of accepting an uphill move of a given magnitude. During the course of the optimisation the temperature is reduced, and so is the associated probability, until both attain low enough values for the procedure to effectively reduce to a very slow descent, at which stage it can be stopped. SA draws obvious analogies to the physical process of annealing, as it originates from simulation studies in this field [Metropol53, Kirkpatr83].

At a given temperature t all n points in the configuration are considered one at a time. A move is generated by adding a deviate from the Cauchy distribution to each coordinate of point $\boldsymbol{x}_i$:

$$\tilde{x}_{ik} = x_{ik} + t \tan p \tag{3.25}$$

where p is drawn uniformly from $(-\pi/2, \pi/2)$, and temperature t is acting as the scale parameter of the distribution. The update formula (3.25) will produce Gaussian like local displacement most of the time, with an occasional long jump, which allows radical moves to be tried even at low temperatures in a controlled manner. Thus temperature can be reduced quicker, compared to using a deviate from the Gaussian distribution in (3.25), while maintaining the properties of global optimisation [Szu87].

The new value of raw Energy (3.16) can be calculated efficiently since only one point is affected, by maintaining a breakdown of the contribution (3.22) of each point to the overall value, as in Section 3.3.3. Rather than calculating the acceptance probability from (3.24), and comparing it with a uniform random number $p \in [0, 1]$ to decide whether to accept the move, the maximum acceptable value of (3.22) can be calculated in advance by rearranging (3.24):

$$\xi = \sigma_{Er}(\boldsymbol{x}_i) - t \ln p \tag{3.26}$$

allowing the calculation of $\sigma_{Er}(\tilde{\boldsymbol{x}}_i)$ to be aborted as soon as it exceeds ξ [Wright89]. This modification improves the performance, especially when the rejection ratio is high. There is no need to randomise the order of points, as in Section 3.3.2, since not all the moves will be accepted according to (3.26), and thus the sequence of accepted moves will automatically be scrambled.

Central to the design of a simulated annealing algorithm is the choice of a *cooling schedule*, i.e. the manner in which the temperature parameter t is reduced. It is a common observation that most of the useful work is done in the middle of the schedule [Dowsland95], and thus it is important to chose an appropriate range of temperatures, over which the cooling occurs. However, the optimal temperature range is not only application dependent, but will be different for every instance of the problem, in general. The actual temperature reduction function is less important, and a geometric function $t_{i+1} = at_i$ is the most common choice in practice, for some $a \in (0,1)$ (*SA.geometric* parameter) [Dowsland95].

Since the amount of useful work done by the algorithm is the overriding consideration, we decided to specify a target acceptance ratio $\bar{\mu}$ (*SA.acceptance* parameter) instead of prescribing the temperature range [Dowsland93]. The actual acceptance ratio μ is the proportion of both downhill and uphill moves accepted to the total number of moves n at each temperature. At high temperature many moves generated by (3.25) will result in a significant increase to (3.22), and according to formula (3.26) will be rejected, thus μ will be low. Conversely, at lower temperature more moves will be downhill, and the remaining uphill ones are likely to be more local, and cause smaller increases to (3.22), therefore, μ will be higher.

Temperature is updated according to the following formula:

$$t_{i+1} = a^{(\bar{\mu}-\mu_i)n} t_i \tag{3.27}$$

If temperature t_i is too high μ_i will be less than $\bar{\mu}$, and for each deficit acceptance temperature is multiplied by a, and thus reduced in proportion to the deficit. Conversely, too low a value of t_i causes μ_i to be greater than $\bar{\mu}$, and the temperature is divided by a for each surplus acceptance, i.e. proportionally increased. Therefore, this schedule will eventually converge at iteration j to a state where $\mu_k \approx \bar{\mu}$, for all subsequent iterations $k \geq j$. The speed of this convergence is dependent on the value of a and t_0 (*SA.initialtemperature* parameter), although we expect the algorithm to be fairly insensitive to their exact choice, as only the value of $\bar{\mu}$ determines the equilibrium temperature. The algorithm goes through a set number of temperature updates (*SA.iterationlimit* parameter), and maintains a record of the best solution generated.

3.4 Experimental Setup

In order to evaluate the relative merits of each algorithm described in Section 3.3, we used a test bed consisting of 66 data collections: 2 dissimilarity matrices, 2 image collections, 6 graphs, and 56 data tables (see Section B.1 and B.2 in the appendices for details). The number of objects in a collection ranged from 9 to 1484, with the median value of 159. Since least-squares MDS algorithms are heuristics for function minimisation, and can only be expected to find local minima in general, we decided to scale each data collection 10 times, to give us a more complete picture of the algorithms' performance. We recorded the minimum and average Energy (3.13) values of 10 trials, and the mean running time per trial for every combination of data collection and algorithm. PCO is an analytical method (see Section 3.3.1), and only a single trial per data collection was necessary.

Parameters of the algorithms were fixed to the following values:

- $p = 2$, dimensionality of MDS configurations
- *NR.epsilon* = *TS.epsilon* = 0.001, chosen on the basis that such a level of accuracy is sufficient for visualisation, i.e. on the order of 1000×1000 pixels

- *GA.iterationlimit* = *ProxScal.iterationlimit* = *SA.iterationlimit* = 1000
- *NR.buffersize* = 100, corresponds to the choice of *iterationlimit*
- *GA.sigma* = 0.01, *GA.epsilon* = 10^{-6}, determined after some informal experimentation, which demonstrated equivalent performance over a broad range of values
- *GA.populationsize* = 32, a compromise between diversity of search for a global minimum and running time; population size as small as 30 has been reported to be adequate for many problems [Reeves95]
- *ProxScal.rate* = 10^{-12}, this level of accuracy was used in collecting Energy measurements for all algorithms
- *SA.geometric* = 0.99, a typical choice amongst SA practitioners [Dowsland95]
- *SA.initialtemperature* = 1.0, well above the equilibrium temperature for all data collections in the test bed.
- *SA.acceptance* = 20%, this value was determined through a formal experiment. As it is argued in Section 3.3.6, the acceptance ratio $\bar{\mu}$ strongly determines the cooling schedule, and thus the performance of the algorithm. *SA* with $\bar{\mu} \in \{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$ has been applied to the test bed to decide where the optimal range lies. Table 3.1 shows the results of the two-tailed Wilcoxon signed-rank test [Siegel56] for pairs of adjacent choices of $\bar{\mu}$, when considering the minimum and average Energy achieved over 10 trials. The parametric t test could be used with the experimental measurements instead. It is slightly more powerful, but at the expense of making restrictive assumptions about the distribution of the measurements. The choice of a non-parametric test over a parametric alternative is discussed also in Section 3.5.

  The null hypothesis H_0 is that minimum Energy for both acceptance ratios under consideration is equivalent, overall. At the outset $p = 0.05$ significance level was assumed, on the basis of which H_0 for $(5\%, 10\%)$, $(10\%, 15\%)$, and $(15\%, 20\%)$ is rejected. The direction of the differences can be decided from the relevant rank sums in Table 3.1(a), and it can be seen that Energy is the lowest at 20%. There is no evidence to reject H_0 for $(20\%, 25\%)$ and $(25\%, 30\%)$, implying that these acceptance ratios are tied in terms of minimum Energy. According to Table 3.1(b) the average Energy achieved at 30% is significantly worse than at 25%, however. At 20 and 25% measurements are tied again, alongside that for 15%. Therefore, the optimal value of $\bar{\mu}$ falls between 20 and 25%, and we decided to use $\bar{\mu} = 20\%$ without further experiments, as we expected the performance across this range to be equivalently good.

Table 3.1: Wilcoxon test results for *SA.acceptance* parameter

relationship[a]	# cases[b]	rank sum[c]	p [d]
10 $\downarrow$ 5	53	1938	0.001
10 $\uparrow$ 5	13	273	
15 $\downarrow$ 10	49	1868	0.001
15 $\uparrow$ 10	17	343	
20 $\downarrow$ 15	43	1599	0.001
20 $\uparrow$ 15	23	612	
25 $\downarrow$ 20	39	1384	0.076
25 $\uparrow$ 20	27	827	
30 $\downarrow$ 25	36	1109	0.985
30 $\uparrow$ 25	30	1102	

(a) minimum Energy

relationship	# cases	rank sum	p
10 $\downarrow$ 5	50	1861	0.001
10 $\uparrow$ 5	16	350	
15 $\downarrow$ 10	41	1564	0.003
15 $\uparrow$ 10	25	647	
20 $\downarrow$ 15	32	1157	0.746
20 $\uparrow$ 15	34	1054	
25 $\downarrow$ 20	30	1151	0.775
25 $\uparrow$ 20	36	1060	
30 $\downarrow$ 25	27	778	
30 $\uparrow$ 25	39	1433	0.036

(b) average Energy

[a]$t \downarrow u$ denotes that Energy for acceptance ratio t% is lower than for u%, and $t \uparrow u$ implies the opposite

[b]the number out of 66 cases for which the relationship holds

[c]absolute differences in Energy for both values of the acceptance ratio parameter across all cases are ordered on their magnitude. Ranks for which differences are negative are summed separately from the positive ones, and their total is $\sum_{r=1}^{66} r = 2211$. These two sums should be roughly equal under H_0: no overall difference in Energy for both acceptance ratios

[d]probability of rejecting H_0 when it is true, nominally an acceptable value – the significance level of the test – is no greater than 5%

Table 3.2: Friedman test results for minimum and average Energy

algorithm	minimum	average
NR	2.86	3.27
TS	2.29	2.80
GA	4.11	3.89
SA	3.27	2.67
IM	2.55	2.61
PCO	5.92	5.76
χ_r^2 [a]	171.38	137.33
W [b]	0.519	0.416

[a] χ_r^2 is Friedman's chi-square statistic; the critical value at $p = 0.001$ level is 20.52

[b] W is Kendall's coefficient of concordance; it ranges between 0 (no agreement between cases) to 1 (complete agreement); the critical value at $p = 0.001$ level is 0.062

The figures in rows 2–7 are mean ranks across 66 cases, which are computed by summing the relative order of minimum or average Energy for a particular algorithm over all cases, and dividing by their total number

3.5 Statistical Analysis

The statistical analysis of MDS algorithms of Section 3.3 is structured into three distinct parts: minimum Energy, average Energy, and running time experiments, each of which consists of 6 related measurements, one for each algorithm, made for every data collection in the test bed. For such an experimental design there are two statistical tests: two-way analysis of variance, or its non-parametric cousin, the Friedman test [Siegel56]. The former can only be used if measurements are independently drawn from normally distributed populations, these populations have identical variance, and their means are linear combinations of effects due to differences in algorithms and data collections. The latter test is more general and robust, as it does not make any of these assumptions, and we preferred to use it for our statistical analysis.

3.5.1 Minimum Energy

A Friedman test has been carried out with the null hypothesis H_0 of no difference between algorithms with regard to minimum Energy, and the alternative hypothesis H_1 of at least one algorithm having a different ranking from the others; the results are presented in the second column of Table 3.2. The values of both Friedman's chi-square statistic and Kendall's coefficient of concordance are statistically significant at a level exceeding 0.001, which allows H_0 be rejected in favour of H_1. This permits the use of the Tukey Multiple Comparison procedure to test for significant differences between algorithm rankings in pairs [Keselman77].

Table 3.3(a) demonstrates that *PCO* has the highest overall ranking, meaning

Table 3.3: Tukey Multiple Comparison results

rank comparison	Q [a]	p [b]
PCO>TS	15.79	* [c]
PCO>IM	14.64	*
PCO>NR	13.32	*
PCO>SA	11.51	*
PCO>GA	7.90	*
GA>TS	7.90	*
GA>IM	6.74	*
GA>NR	5.43	*
GA>SA	3.62	0.003
SA>TS	4.28	*
SA>IM	3.13	0.010
SA>NR	$\ll$ 4.03	0.244
NR>TS	2.47	0.008
NR>IM	$\ll$ 4.03	0.037
IM>TS	$\ll$ 4.03	0.089

(a) minimum Energy

rank comparison	Q	p
PCO>IM	13.69	*
PCO>SA	13.42	*
PCO>TS	12.83	*
PCO>NR	10.79	*
PCO>GA	8.09	*
GA>IM	5.59	*
GA>SA	5.33	*
GA>TS	4.74	*
GA>NR	2.70	0.103
NR>IM	2.90	0.005
NR>SA	$\ll$ 4.03	0.003
NR>TS	$\ll$ 4.03	0.045
TS>IM	$\ll$ 4.03	0.165
TS>SA	$\ll$ 4.03	0.159
SA>IM	$\ll$ 4.03	0.439

(b) average Energy

[a] Q is Tukey's statistic for multiple comparisons; the critical value at $p = 0.05$ level is 4.03
[b] the p value associated with a one-tailed Wilcoxon signed-rank test
[c] * denotes a significant Tukey comparison, and thus no need for a Wilcoxon test

that it achieves the worst minimum Energy. The comparisons between *GA* and the remaining algorithms, except *SA*, are significant at $p < 0.05$ level. To verify the tie of *GA* and *SA*, we decided to perform a one-tailed Wilcoxon signed-rank test, because it is considerably more powerful [Siegel56]. It produces a highly significant p value of 0.003 (see Table 3.3(a)), which taken together with the multiple comparison results implies that *GA* is the second worst in terms of minimum Energy. The only remaining significant Tukey comparison is that between *SA* and *TS*; additionally, the Wilcoxon test is significant for (*SA*,*IM*), (*NR*,*TS*), and (*NR*,*IM*) pairs. Thus, it can be inferred that *IM* and *TS* are tied in the first place, and are significantly better than both *SA* and *NR*, which in turn are also equivalent in performance.

3.5.2 Average Energy

The third column of Table 3.2 contains the results of the Friedman test with the null hypothesis H_0 of no difference between algorithms with regard to average Energy, and the alternative hypothesis H_1 of at least one algorithm having a different ranking from the others. Both Friedman's chi-square statistic and Kendall's coefficient of concordance are statistically significant at $p < 0.001$ level, which allows H_0 be rejected in favour of H_1. The Tukey Multiple Comparison procedure can subsequently be performed, to find out which pairs of algorithm rankings are significantly different.

Since PCO is an analytical method (see Section 3.3.1) the minimum and average values of Energy are taken to be the same. The Tukey comparisons between *PCO* and the other algorithms are significant at the 5% level according to Table 3.3(b), implying that *PCO* has the highest overall ranking with regard to average Energy. The multiple comparisons between *GA* and the remaining algorithms are significant at $p < 0.05$ level, except with *NR*, and this tie is confirmed by the Wilcoxon test. All remaining Tukey comparisons are not statistically significant; however, the Wilcoxon tests of *NR* versus *IM*, *SA*, and *TS* in turn are significant at the 5% level. Thus the following ordering can be inferred: *SA*, *IM*, and *TS* jointly achieve the lowest average Energy, and are significantly better than *NR* and *GA*, which are tied themselves.

3.5.3 Running Time

The algorithms under test have different time complexities: *PCO* is $O(n^3)$ as it is based on an eigendecomposition of an $n \times n$ matrix [Press92] (also see Section 3.3.1 and 4.3), the other algorithms are least-squares techniques, and are $O(n^2)$ per iteration[§], except that *IM* is $O(n^3)$ per iteration when minimising raw Energy (3.16), as it involves a multiplication of two $n \times n$ matrices [deLeeuw77, Borg97]. Therefore, we decided not to perform the analysis of variance on the time measurements,

[§]follows from the fact that the whole dissimilarity matrix $\boldsymbol{\Delta}$ of order $n \times n$ has to be inspected at every iteration, where n is the number of objects

Table 3.4: Comparison of algorithm running time

algorithm	total time[a]	time ratio[b]	linear fit[c]
SA	136	n/a	n/a
NR	152	1.12	1.34
PCO	165	1.22	1.51
IM	345	2.54	2.74
TS	1074	7.91	9.48
GA	86696[d]	638.36	764.03

[a]average time in minutes required for a single test trial with the test bed
[b]ratio of average time for a given algorithm to that of *SA*
[c]the parameter of a least-squares fit of *SA* timings to that of each algorithm
[d]approximately 60 days

as the majority of data sets in the test bed are small (see Section 3.4), and an unfair advantage would be given to *PCO* and *IM*, which are efficient for such data, but are quickly outperformed when n grows.

The second column of Table 3.4 reports the average running time of a single test trial for each algorithm. The fastest algorithm with our test bed is *SA*, and it is closely followed by *NR* and *PCO*. For convenience, the ratio of running time for every algorithm to that for *SA* is presented in the third column of Table 3.4. *IM* and *TS* are considerably slower, but are within one order of magnitude of *SA*. *GA* is by far the slowest algorithm, as it required almost one and a half years of CPU time of an Intel Pentium II 300 MHz workstation to complete the experiment[¶].

The total running time and its ratios in Table 3.4 are peculiar to the composition of the test bed. To derive a more robust estimation of relative speed of the algorithms, we decided to perform a linear regression of measurements for *SA* to that of other algorithms. The timings for an individual algorithm i are collected in the vector $\boldsymbol{m}^{(i)}$, with one element for each data collection in the test bed. The parameter a_i that gives the best least-squares fit of the model $\boldsymbol{m}^{(i)} = a_i \boldsymbol{m}^{(SA)}$ is taken as an empirical estimate of how much slower algorithm i is compared to *SA*, and is reported for each algorithm in the last column of Table 3.4. These parameters are in a good agreement with the running time ratios, and thus the conclusions of the previous paragraph hold.

3.5.4 Identifying the Best Algorithm

SA was identified as the fastest algorithm for our experimental settings, while at the same time sharing the lowest average and second lowest minimum Energy ranking. Close contenders are *IM* and *TS*, which achieved the lowest minimum and

[¶]actually a number of workstations have been used, and the timings normalised to that of the 300MHz CPU

Table 3.5: Wilcoxon test results for *SA-2500* and *hybrid* algorithms

	SA-2500			hybrid		
algorithm	#cases[a]	rank sum[b]	p [c]	#cases	rank sum	p
PCO	65	2166	0.001	65	2209	0.001
GA	47	1670	0.001	59	2057	0.001
SA-1000	53	1773	0.001	66	2211	0
NR	30	1203	0.538	47	1623	0.001
IM	34	991	0.469	44	1374	0.001
TS	25	897	0.185	44	1481	0.007
SA-2500		n/a		46	1624	0.001

[a]the number out of 66 cases for which minimum Energy achieved by a given algorithm was greater than that of *SA-2500* or *hybrid*

[b]absolute differences in minimum Energy for both algorithms across all cases are ordered on their magnitude. The sum of ranks for which *SA-2500* or *hybrid* achieved lower Energy than a given algorithm is reported; the rank total is $\sum_{r=1}^{66} r = 2211$. The rank sum should be about a half of the total under H_0: no overall difference in minimum Energy for both algorithms

[c]probability of rejecting H_0 when it is true, nominally an acceptable value – the significance level of the test – is no greater than 5%

average Energy, however at a 2.5 and 8 fold increase in running time, respectively (see Table 3.4). This comparison may be unfair on *SA*, which exhibits good global optimisation characteristics, confirmed by the average Energy performance, but perhaps was not given enough time to refine its solutions, and achieve competitive minimum Energy scores. Therefore, we decided to re-run the *SA* tests, setting *SA.iterationlimit* to 2500 this time (see Section 3.3.6 and 3.4).

Table 3.5 shows the results of the two-tailed Wilcoxon signed-rank test between *SA-2500* and all other algorithms. The null hypothesis is H_0: no difference in minimum Energy between *SA-2500* and a given algorithm. H_0 is rejected for *PCO*, *GA*, and *SA-1000*; moreover, *SA-2500* can be identified to be significantly better that these algorithms from the relevant rank sums. There is no evidence to reject H_0 for *NR*, *IM*, and *TS*, however. Since *SA-1000* was tied with *NR*, too, but worse than the other two algorithms in terms of minimum Energy (see Section 3.5.1), *SA-2500* is only slightly better, and occupies the middle ground.

3.5.5 Hybrid Algorithm

Section 3.5.4 demonstrated that prolonging the *SA* run did improve the minimum Energy performance, but not as much as we hoped. We realised that the extra time could be spent on running a different heuristic after *SA-1000*, instead. *IM* seemed the best choice, as it is a strictly local minimisation procedure, i.e. will

Table 3.6: Linear regression comparison of MDS algorithms

algorithm	ratio[a]
PCO	1.9873
GA	1.0561
SA-1000	1.0476
NR	1.0446
SA-2500	1.0397
IM	1.0018
TS	1.0016

[a]overall ratio of minimum Energy for a given algorithm to *hybrid*

converge to the local minimum nearest to an *SA-1000* solution[||]. Therefore, the best configuration of 10 *SA-1000* runs for each data collection in the test bed was used as a starting point for *IM*, and the final value of Energy (3.13) recorded. The total time of the test for this *hybrid* algorithm was 2.87 times that of *SA-1000* alone. This is comparable to 2.99 for *SA-2500*, and 2.54 for *IM* (see Table 3.4). The linear regression coefficients (see Section 3.5.3) are also similar at 3.01, 2.99, and 2.74 respectively. Thus, the best algorithm can be decided on the minimum Energy performance alone.

The results of the two-tailed Wilcoxon signed-rank test between *hybrid* and the remaining algorithms can be found in Table 3.5. The null hypothesis H_0 of no difference between algorithms can be rejected in every case. The rank sums are in favour of *hybrid*, thus it can clearly be identified as the best MDS algorithm. As expected, *hybrid* improves on *SA-1000* configurations for all data collections in the test bed.

To complete the comparison of the various algorithms described in this chapter, we decided to perform a linear regression of minimum Energy measurements for the best algorithm to that of the others. Assuming that measurements for all data collections in the test bed with a given algorithm i are collected in the vector $\boldsymbol{m}^{(i)}$, the linear model becomes: $\boldsymbol{m}^{(i)} = a_i \boldsymbol{m}^{(hybrid)}$. Table 3.6 presents all coefficients a_i in the descending order. This ordering is exactly the same as inferred by the statistical tests, namely: *PCO* is the worst in terms of minimum Energy, followed by *GA*, the next place is jointly occupied by *NR* and *SA-1000*; *SA-2500* is only slightly better. *TS* and *IM* achieve considerably lower Energy overall, and are only 0.2% worse than *hybrid* on average. The improvement between *SA-1000* and *hybrid* is almost 5%, and fully justifies using a local minimisation heuristic to refine *SA* solutions.

[||]*IM* is a deterministic algorithm, thus for a given starting configuration it will always converge to the same local minimum

3.6 Qualitative Evaluation

In Section 3.5 we were able to demonstrate by means of statistical tests that the algorithms of Section 3.3 differ significantly in terms of output quality. It is a much more objective and efficient method of comparing MDS configurations for all 6×66 combinations of an algorithm and a test data collection, than inspecting them visually. Nonetheless, it is important to show some examples, so that the magnitude and character of the differences can be appreciated. For this purpose we have selected a sample data set from the test bed, which appears in Section B.1 in the appendices as *cpu-performance*. It is a tabular record of 7 quantitative attributes for 209 CPUs.

Visualisations produced by all algorithms are arranged side by side in Figure 3.4. The experimental setup of Section 3.4 has been used, thus the best configuration of 10 runs for each algorithm is shown, except that the *PCO* configuration is unique. To ease the task of comparing these visualisations, and only focus on significant differences, a common orientation of the figures was assumed. Each configuration has been transformed by Procrustes Analysis [Borg97, Cox94] with regard to a reference configuration – that for the *hybrid* algorithm, as it achieved the lowest value of Energy (3.13) of all algorithms. Procrustes Analysis seeks the optimal transformation of a configuration to match a given target configuration. Since Energy (and Stress (3.11) for that matter) with the ratio MDS model are invariant under dilation and rigid motions: translation, rotation, and reflection of a given configuration, these transformations were allowed.

As can be seen from Figure 3.4(e), the configurations for *SA* and *hybrid* – the reference configuration – are very similar. Since *hybrid* consists of an *SA* run followed by *IM*, i.e. it finds the local minimum closest to the configuration produced by *SA*, such a correspondence was expected.

The level of agreement between a pair of configurations $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be measured by the Tucker's congruence coefficient [Tucker51] (also see Section 3.2.2):

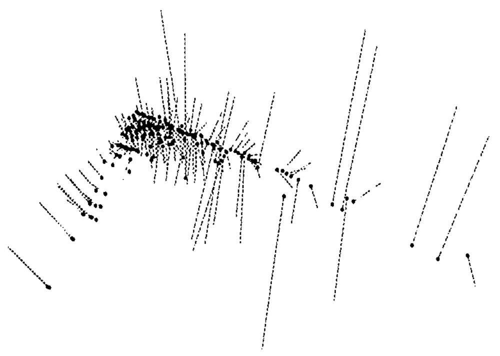
$$c(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\sum_{i<j} d_{ij}(\boldsymbol{X}) d_{ij}(\boldsymbol{Y})}{\left(\sum_{i<j} d_{ij}^2(\boldsymbol{X})\right)^{\frac{1}{2}} \left(\sum_{i<j} d_{ij}^2(\boldsymbol{Y})\right)^{\frac{1}{2}}} \tag{3.28}$$

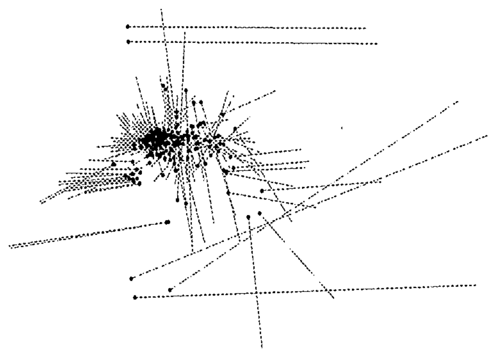
The coefficient achieves its maximum value 1 when $d_{ij}(\boldsymbol{X}) = b d_{ij}(\boldsymbol{Y})$, for all i, j. The congruence coefficient is invariant under dilations of $\boldsymbol{X}$ and $\boldsymbol{Y}$:

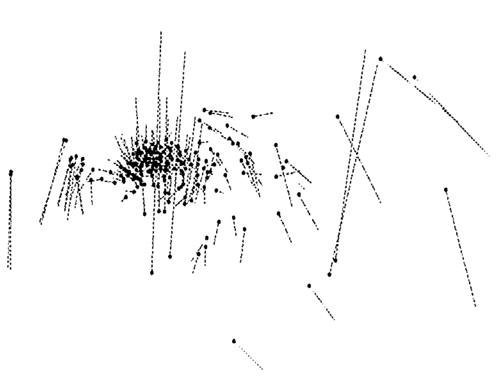
$$c(a\boldsymbol{X}, b\boldsymbol{Y}) = \frac{\sum_{i<j} a d_{ij}(\boldsymbol{X}) b d_{ij}(\boldsymbol{Y})}{\left(\sum_{i<j} a^2 d_{ij}^2(\boldsymbol{X})\right)^{\frac{1}{2}} \left(\sum_{i<j} b^2 d_{ij}^2(\boldsymbol{Y})\right)^{\frac{1}{2}}} = c(\boldsymbol{X}, \boldsymbol{Y})$$

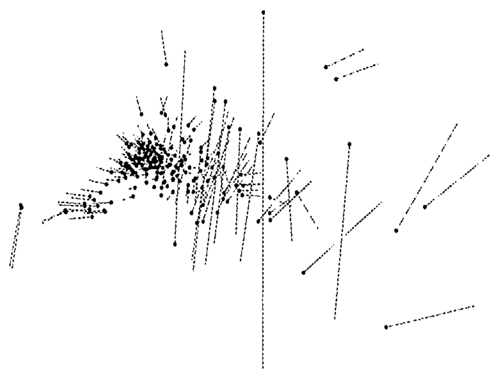
and under rigid motions, since distances are not affected by them. Therefore, $c(\boldsymbol{X}, \boldsymbol{Y})$ will not change after carrying out Procrustes Analysis between configurations $\boldsymbol{X}$ and $\boldsymbol{Y}$, and determines their fit in advance.

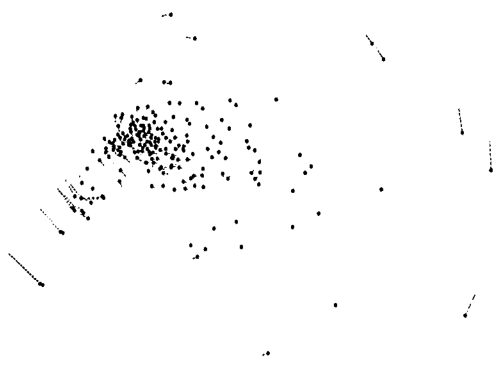
Let the configuration for algorithm z be denoted as $\boldsymbol{X}^{(z)}$. The value of $c(\boldsymbol{X}^{(SA)}, \boldsymbol{X}^{(hybrid)})$, which can be found in Table 3.7, is higher than for any other pair, and very close to the maximum value possible. Overall, $\boldsymbol{X}^{(IM)}$ in Figure 3.4(f)


(a) $\sigma_E(PCO) = 0.0746$


(b) $\sigma_E(GA) = 0.0475$


(c) $\sigma_E(NR) = 0.0418$


(d) $\sigma_E(TS) = 0.0318$


(e) $\sigma_E(SA) = 0.0327$

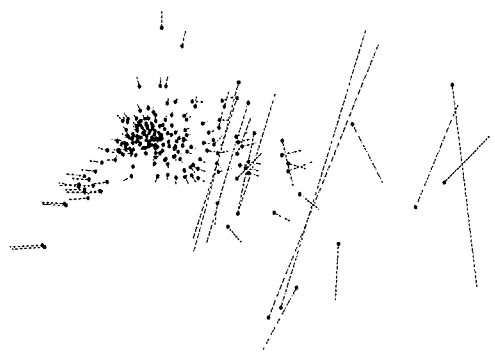

(f) $\sigma_E(IM) = 0.0325$

Figure 3.4: Visualisations of the *cpu-performance* data table with different MDS algorithms. Dots represent individual CPUs – rows in the table. Each configuration was fitted to the one for the *hybrid* algorithm by Procrustes analysis; $\sigma_E(hybrid) = 0.0308$. Lines link pairs of corresponding points in both configurations, and thus show the extent of the differences

Table 3.7: Congruence coefficient for pairs of MDS configurations

	hybrid	SA	IM	TS	NR	GA
SA	0.9989					
IM	0.9946	0.9938				
TS	0.9918	0.9905	0.9923			
NR	0.9909	0.9881	0.9897	0.9921		
GA	0.9734	0.9725	0.9733	0.9748	0.9736	
PCO	0.9783	0.9765	0.9800	0.9796	0.9796	0.9604

corresponds well to $\boldsymbol{X}^{(SA)}$, except that some outliers assume an opposite orientation with regard to the main group of points. Nonetheless, Energy (3.13) is virtually the same in both cases, and the congruence coefficient is very high among these two configurations and also $\boldsymbol{X}^{(hybrid)}$. This is an apt demonstration that there can exist equivalently good proximity visualisations of a given data collection, with nontrivial differences between them. This point is emphasised when considering $\boldsymbol{X}^{(TS)}$, which exhibits even more differences to $\boldsymbol{X}^{(hybrid)}$, but is judged similar by both the loss function and the congruence coefficient.

$\boldsymbol{X}^{(NR)}$ is a significant departure from the reference configuration, and consequently congruence with $\boldsymbol{X}^{(SA)}$ and $\boldsymbol{X}^{(IM)}$ is moderate, relative to the values of coefficient c for the algorithms considered so far. However, $c(\boldsymbol{X}^{(NR)}, \boldsymbol{X}^{(TS)})$ is the fifth highest, despite *NR* being much worse in terms of Energy, and many similarities between Figures 3.4(c) and 3.4(d) can be noticed.

PCO and *GA* have produced very different configurations to that of the *hybrid* algorithm, since many long lines can be seen in Figure 3.4(a) and 3.4(b). Values of the congruence coefficient with configurations for the other algorithms are relatively low, and $c(\boldsymbol{X}^{(PCO)}, \boldsymbol{X}^{(GA)})$ is actually the lowest. Energy of $\boldsymbol{X}^{(PCO)}$ is much worse than for $\boldsymbol{X}^{(GA)}$, which in turn is inferior to the remaining configurations. The *PCO* configuration is completely different to that of the least-squares algorithms. Points are close to two perpendicular lines – principal components (see Section 3.3.1), and local detail is not represented as well as with the other algorithms.

3.7 Discussion

Several MDS algorithms have been described in this chapter, and compared on the basis of their performance with a sample of small and medium-sized data collections. A histogram of Energy (3.13) minima calculated by the *hybrid* algorithm for all data sets is presented in Figure 3.5. The distribution of data collections between buckets is typical of a balanced sample, with most having moderate Energy, and extreme cases being in minority. The minimum Energy for a data set is

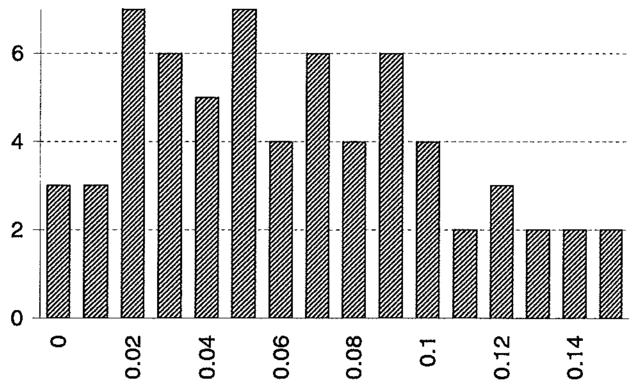


Figure 3.5: Histogram of Energy minima calculated by the *hybrid* algorithm for all collections in the test bed

indicative of its complexity, and a challenge it represents to an MDS algorithm in finding a good low-dimensional representation for it. Therefore, we are confident in the validity of statistical inference presented in Section 3.5, and conclude that these results go beyond just the data collections considered there.

PCO is shown to achieve inferior results both objectively and subjectively, and this can be explained by the characteristics of the loss function (3.18) it actually minimises, compared to Energy (3.13). However, its solutions are unique, and can be computed efficiently for smaller data sets; consequently, with such data *PCO* might serve for other MDS algorithms as the source of a better starting configuration than a purely random one [Kruskal78b]. The least-squares algorithms are much more effective at minimising Energy. The overall difference in Energy between the best and the worst heuristic is less than 6%; thus the correctness of these algorithms is cross-validated.

It is not practical to run *GA* on standard computer hardware, as it takes prohibitively long to produce its solutions. However, this algorithm can easily be parallelised [Reeves95], and merits consideration with such an architecture. *NR* and *SA* are the fastest MDS algorithms, and actually perform slightly better than *GA*. *IM* and *TS* are able to achieve even lower Energy on average, but at the same time are significantly slower. If one is willing to accept such an increase in running time then a combination of *SA* followed by *IM* performs the best. Such a hybrid approach takes advantage of the global optimisation characteristics of Simulated Annealing, in order to find the neighbourhood of a global or a good local minimum, and converge on this minimum with a descent technique of local optimisation.

3.8 Related Work

Multidimensional Scaling has been applied in many other areas, and consequently the literature is vast and dispersed over many periodicals and books. We will not attempt to give an overview of the developments to date, and refer the reader elsewhere [Borg97, Cox94]. Also, the Psychometrika journal is the single richest source of reference for MDS, as it regularly includes articles on the subject. Here, we give a brief summary of other work directly related to the algorithms discussed in this chapter.

The graph drawing community proposed a number of algorithms for constructing straight-line drawings of general undirected graphs, and an evaluation of the five most popular has been carried out [Brandenb95]. The study compared running time, number of edge crossings, and edge length uniformity across a collection of 79 graphs. One of the algorithms [Kamada89] is in fact an MDS implementation minimising raw Energy (3.16) (see Section 3.3.2). This algorithm was found to produce drawings of comparable quality to that of other algorithms, with the least computational cost. This favourable result should, therefore, extend to other MDS algorithms for this domain. The analysis presented (in the form of charts) was inconclusive for the aesthetic criteria, and the recommendations were based purely on running time of the algorithms.

In Sections 3.3.6 and 3.5 it has been demonstrated that the Simulated Annealing heuristic improves the likelihood of finding a global minimum of a loss function like Energy (3.13). The Deterministic Annealing approach [Klock97] capitalises on the benefits of its stochastic counterpart, while avoiding the multitude of choices and challenges in designing and fine tuning a successful algorithm. It does so by considering the statistical behaviour of Simulated Annealing as an entropy maximisation procedure, and deriving an approximate but computationally tractable solution. An algorithm for minimising Stress (3.11) based on this method is subsequently proposed.

Chapter 4

Large Scale Visualisation

Abstract data collections can be visualised effectively with Multidimensional Scaling; however, the process becomes very time consuming for large data sets. This chapter explores alternatives that aim to achieve a substantial speed up, without sacrificing the quality of visualisation too much. A novel technique is proposed that combines cluster analysis with least-squares MDS, to reduce both running time and space requirements of traditional MDS, while adequately representing proximity relationships in a large data collection. Multivariate statistics provides an alternative method of visualising data tables, which is equivalent to classical scaling, but dependent on the number of attributes instead of objects, and thus more efficient for large data sets. Both methods are compared statistically and qualitatively on a sample of large data tables.

4.1 Challenges of Visualising Large Data Collections

The amount of screen space available to each icon representing an object from a data collection becomes very limited if the collection is large. Here, it is more important to convey a proper overview of relationships among the objects than individual detail. Proximity Visualisation is particularly suitable, as it will accurately portray the distribution of objects and clusters. Detailed information can be accessed by interacting with this overall visualisation. For example, an object could be singled out, and its attributes shown to the user. Alternatively, a subset of the objects could be specified with a visual query, and visualised on its own with more detailed icons, as in Chapter 2.

Traditional multidimensional scaling is unsuitable for visualising a large data collection, because the amount of pairwise dissimilarities that have to be considered grows quadratically in the number N of objects (see Section 3.1). However, there is a degree of redundancy in collecting all $\binom{N}{2}$ dissimilarities, especially when no error is present [Young72], as is the case with the dissimilarity coefficients of Section 1.2. Thus we may attempt to identify and discard unimportant dissimilarities without affecting the accuracy of the visual representation.

Let us define the number of *degrees of freedom* for configuration $\boldsymbol{X}$ of N points in p dimensions [Young70]:

$$dof(\boldsymbol{X}) = p(N-1) - \frac{p(p-1)}{2} \tag{4.1}$$

A simple relation has been empirically shown to exist between $dof(\boldsymbol{X})$ and the proportion λ of dissimilarities that must be retained for an MDS procedure to successfully recover a matching configuration $\boldsymbol{X}$ [Spence74]:

$$\begin{aligned} \frac{\lambda\binom{N}{2}}{dof(\boldsymbol{X})} &> 3 \\ \lambda &> \frac{6p(N-1) - 3p(p-1)}{N(N-1)} \\ \lambda &> \frac{6p}{N}, \text{ for } N \gg p \end{aligned} \tag{4.2}$$

Admittedly, the study is only based on $N \in \{32, 40, 48\}$ and $p = 3$, but shows that there is scope for discarding some dissimilarities without compromising the recovery of $\boldsymbol{X}$.

In practice, most large data collections are multivariate tables, for example the test data of Section B.3 in the appendices. Dissimilarity matrices with a comparable number of objects are rare, as such a huge number of pairwise experimental measurements would be impractical to obtain, except when derived from an intermediate representation, e.g. some sort of summary for each object, which may be expressed as a multivariate table, anyway. Alternatively, an incomplete design could be applied, where only a subset of all dissimilarities is collected [Spence74], adhering to the guideline provided by (4.2) when deciding on the subset size λ. Likewise, large connected graphs are not very common, and disconnected graphs can be visualised by considering each connected component individually.

The advantage of considering only tabular data is that the variables (columns) can be manipulated to derive a smaller number of variables, preferably two or three to facilitate direct visualisation. Attribute clustering is one possibility, whereby correlated groups of variables are hierarchically aggregated, starting with as many groups as the original variables [Anderber73]. A particular level in the hierarchy with the desired number of groups is then selected, and these groups serve as the new variables. However, a more fine-grained composition of the original variables is possible with Principal Components Analysis [Pearson01, Hotellin33]. With this reclassification method, each derived variable is expressed as a linear combination of the original variables, subject to the new variables being mutually uncorrelated, or orthogonal in other words.

4.2 Incremental Multidimensional Scaling

An *incremental MDS* method, that achieves a good compromise between solution quality and the number of dissimilarities considered, is proposed here. The discarded dissimilarities are those which are not needed to preserve the overall shape of the MDS configuration. This overall shape is established by first carrying out a single-link clustering of the objects from a collection using a suitable dissimilarity coefficient (see Section 1.2). The clustering is used to select a subset of objects, which make up cluster diameters at a certain level of the cluster hierarchy. Standard MDS is then applied to this subset – *full scaling* stage – thus establishing a skeleton of marker points. The remaining objects are positioned within this skeleton configuration by only considering their dissimilarities to the fixed objects – *single scaling* stage. For a very large data collection the skeleton configuration can itself be built up incrementally.

The particular choice of an MDS method has little significance to the overall incremental approach described in the following sections. However, the method should allow new objects to be added to an existing configuration (single scaling) with little cost. Least-squares metric MDS fulfils this criterion; see Section 3.3 for examples of algorithms that could be applied.

4.2.1 Single-link Clustering

To determine an ordering of objects for the incremental method a Minimum Spanning Tree (MST) is computed for a fully connected graph, with one vertex for each object, and edge weights taken to be dissimilarities between vertices. MST is a connected graph with N vertices and $N-1$ edges, having the minimal cumulative weight of its edges [Harary69]. Prim's algorithm [Prim57] is used, as it is the most efficient MST algorithm for dense graphs [Sedgewic92].

There is a close relationship between a single-link clustering and an MST [Rohlf82], and the former can be derived from the latter trivially. Edges of the MST with the greatest weight link the highest level clusters in the cluster hierarchy, and thus define the overall structure of data. The order of objects is determined by taking vertices of MST edges sorted in descending order of weight, with only the first occurrence of a vertex noted. Therefore, the position of an object in the sequence will depend on its structural importance, with the most important objects coming first.

This discussion assumes no significant effect of outliers. In cases where they are a real concern, a data preparation stage might be necessary, for example replacing actual dissimilarities by their rank order (see Sections 1.2.9 and 3.2.1). Other cluster analysis techniques can be used instead, e.g. complete or average linkage; however, single-link clustering is the least computationally intensive [Anderber73].

4.2.2 The Algorithm Outline

1. objects are sorted in a descending Minimum Spanning Tree order $v_1 \ldots v_N$
2. $v_1 \ldots v_{n_1}$ are assigned random coordinates; $i \leftarrow 1$
3. full scaling is performed on $v_1 \ldots v_{n_i}$
4. $v_{n_i+1} \ldots v_{n_{i+1}}$ are added to the configuration by single scaling with respect to $v_1 \ldots v_{n_i}$
5. $i \leftarrow i+1$, and steps 3 and 4 are repeated until there are no more objects to add: $n_i = N$

The choice of n_1 determines how far to go down the single-link cluster hierarchy in order to form the first skeleton of marker points. The configuration is established by performing full scaling starting from random positions. Further objects from the lower levels in the cluster hierarchy up to a total of n_2 are then included in the configuration by single scaling. The extended set of n_2 points is taken as the initial configuration for a further full scaling run, and the new skeleton so formed is enlarged to n_3 points by single scaling. The process of refining to form a skeleton by full scaling and then enlarging the configuration by single scaling is repeated until there are no more objects to be included. Note that the saving in cost arises because at the final step new objects are incorporated using single scaling only. These objects are associated with the lowest levels in the cluster hierarchy, and so may be positioned independently of one another without affecting the overall structure of the configuration.

A sensible choice of n_1 – the size of the initial skeleton – is crucial to the success of the incremental method. The initial full scaling starts from random positions, and if a large skeleton is to be determined the computational cost may be high. On the other hand, if too few objects are included the configuration will not represent the overall structure adequately. It may be advantageous to scale several times from different random starting points and select the configuration with the lowest Energy (3.13), since the computational cost is dominated by later scaling runs, and the quality of the initial skeleton determines the final outcome.

The sequence $n = \langle n_1, \ldots, N \rangle$ of constants is chosen in the following way:

$$\begin{aligned} u_1 &= N \\ u_{i+1} &= u_i^{\alpha} \\ n_i &= u_{k-i+1} \end{aligned} \tag{4.3}$$

where k is the smallest natural number such that $u_k = n_1 \leq \phi$ for some constant ϕ, and $1 \leq i \leq k$. Thus, the number n_1 of objects to be included in the initial skeleton is determined by ϕ, for given N and α: $n_1 = N^{\alpha^{\lceil \log_\alpha \log_N \phi \rceil}}$[†]. The choice of a suitable value for the exponent α is discussed in Section 4.2.3.

[†]calculated by expanding the sequence n: $n_1 = N^{\alpha^{k-1}}$, and solving $\min_k n_1 \leq \phi$ for k

Time Complexity

1.[‡] $O(N^2)$

2. $O(n_1) = O(\phi) = O(1)$

3. $O(n_i^2)$ per iteration, with $O(n_i)$ iterations: $O(n_i^3)$[§]

4. $O(n_i)$ per iteration per object, with $O(n_i)$ iterations: $O((n_{i+1} - n_i)n_i^2) = O(n_{i+1}n_i^2)$

$$\begin{aligned} T(N) &= O(N^2) + \sum_{i=1}^{k-1} O(n_i^3) + \sum_{i=1}^{k-1} O(n_{i+1}n_i^2) = O(N^2) + \sum_{i=1}^{k-1} O(n_{i+1}n_i^2) \\ &= O(N^2) + O(n_k n_{k-1}^2) = O(N^2) + O(N^{1+2\alpha}) \end{aligned} \tag{4.4}$$

Space Complexity

1. $O(N)$, dissimilarities are not stored, but computed on demand

2. $O(1)$

3. $O(n_i^2)$, dissimilarities are cached for efficiency[¶]

4. $O(n_i)$, dissimilarities are cached for efficiency

5. $O(N)$ for the configuration itself

$$\begin{aligned} S(N) &= O(N) + \sum_{i=1}^{k-1} O(n_i^2) + \sum_{i=1}^{k-1} O(n_i) = O(N) + \sum_{i=1}^{k-1} O(n_i^2) \\ &= O(N) + O(n_{k-1}^2) = O(N) + O(N^{2\alpha}) \end{aligned} \tag{4.5}$$

4.2.3 Empirical Characteristics

Table 4.1 shows results of tests run on synthetic data. Each data set consists of 5000 points randomly placed within a 3-, 4-, 5- or 6-dimensional cube, with a unit diagonal to keep the dissimilarity coefficient – Euclidean distance – uniformly in the range [0, 1]. Incremental MDS, based on the NR algorithm of Section 3.3.2, has been applied to represent these data in two dimensions. The initial value of an index, either Energy (3.13) or Stress (3.11), refers to its value after single scaling from 300 to 5000 points. The remaining columns contain values after 100 iterations of full scaling on 5000 points, 200 iterations, and the final value after the algorithm has converged. To circumvent the natural variability of the method,

[‡]labels correspond to steps of the algorithm

[§]this is also the worst case running time of standard MDS: $O(N^3)$

[¶]this is also the space requirement of standard MDS: $O(N^2)$

Table 4.1: Energy and Stress at various stages of optimisation

Energy				
data	initial	100 iterations	200 iterations	final
3D	0.0782655	0.0697988	0.0695496	0.0694058
4D	0.1127830	0.0997244	0.0989172	0.0989170
5D	0.1376181	0.1161879	0.1159678	0.1159672
6D	0.1567037	0.1264447	0.1263261	0.1262402
Stress				
3D	0.0847296	0.0636327	0.0641757	0.0643987
4D	0.1315574	0.0944356	0.0926183	0.0926292
5D	0.1779833	0.1125812	0.1117612	0.1117596
6D	0.2166460	0.1253326	0.1248531	0.1245055

each figure is taken as an average of 10 tests. Because Stress is a different loss function, it is possible for its value to increase when the value of Energy decreases during a minimisation procedure. This effect can be seen for the 3D and 4D tests in the last two columns of Table 4.1.

The results indicate that the initial value of Energy in all cases is already quite low. With 100 iterations it can be brought very close to the final value, i.e. a minimum. Performing a fixed number of iterations of full scaling, for example 100, on all objects, without caching the dissimilarities, would not affect time and space complexities of the algorithm.

In order to suggest a suitable exponent α that determines the sequence (4.3) of constants in the incremental algorithm, extensive testing has been performed on a sample data set, which appears as *nbody-2* in Section B.3 of the appendices. This data set is a snapshot of an astronomical N-body simulation [Tout97] consisting of $N = 14898$ objects with 12 attributes. A range $[0.5, 0.91]$ of exponents has been sampled at intervals of 0.0025. For every exponent α the sequence $n = \langle N^\alpha, N \rangle$ has been used with the incremental version of the NR algorithm (see Section 3.3.2), and the final value of Energy and Stress in two dimensions recorded. The test has been repeated 10 times, and the resulting minimum value of Energy and Stress for every exponent is plotted in Figure 4.1. The chart also includes a corresponding plot of the minimum over the first three measurements for each exponent. Best of 3 and best of 10 plots coincide for the range $[0.65, 0.91]$, which indicates that incremental MDS for large enough exponents is consistent between multiple runs.

It can be seen from Figure 4.1 that the relationship between exponent and indices is negative exponential: $f(\alpha) = a + br^\alpha$, $r < 1$. The α-axis scale is logarithmic in effect, because $t = N^\alpha$ is actually used in the tests. This results in an inverse polynomial relationship between N^α and indices: $f(t) = a + bt^c$, $c = \log_N r < 0$, which implies an even slower decay rate. Therefore, choosing an exponent is a

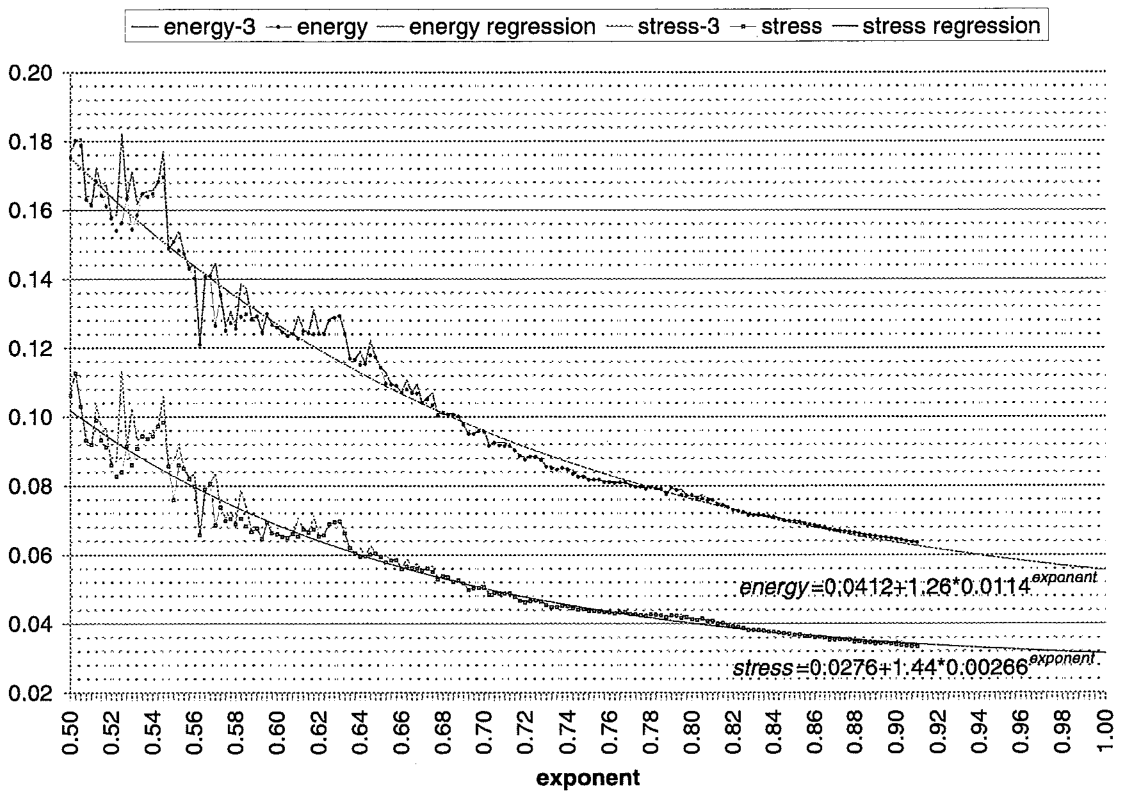


Figure 4.1: Plot of Energy and Stress minima vs. exponent for N-body simulation data

matter of trading off speed for accuracy, with a diminishing advantage for larger exponent values. A value in the range $[2/3, 3/4]$ seems to be a good compromise. The fitted exponential regressions predict the Energy minimum for this particular data set at 0.05551, and Stress at 0.03143. Standard MDS for all 14898 objects has yielded 0.05418 and 0.03283 respectively, which suggests that these regressions are a good fit to the measurements. It is worth noting that this single MDS run takes about a week to compute on an Intel Pentium II 300 MHz workstation, which is enough time to test the range $[0.5, 0.75]$ 10 times. However, to extend the measurements up to $\alpha = 0.91$ requires an additional month of computation.

4.3 Principal Components Analysis

Principal Components Analysis (PCA) is a multivariate statistics method for linearly transforming a sample of N p-variate vectors $\boldsymbol{X} = [x_{ik} : i = 1, \ldots, N; k = 1, \ldots, p]$ into a new sample of q-variate vectors $\boldsymbol{Y} = [y_{il} : l = 1, \ldots, q]$, such that the columns of $\boldsymbol{Y}$ are uncorrelated, and $q \leq p$ [Flury97]. Thus, each of the q derived variables is expressed as a linear combination of p correlated, measured variables [Hotellin33].

The q derived variables are referred to as Principal Components (PCs), and form a system of orthogonal axes. If we consider $\boldsymbol{X}$ to be a configuration of N points $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$ in a p-dimensional Euclidean space then the first PC defines a line through this space that minimises the sum of squared distances of the points from it, and thus maximises the variance of coordinates $\{x_i' = y_{i1}\}$ of an orthogonal projection of $\{\boldsymbol{x}_i\}$ on this line [Pearson01]. The second PC is a line that maximises the variance of the projection coordinates $\{x_i'' = y_{i2}\}$ of the points, subject to being perpendicular to the first PC. Taken together the first two PCs give a plane of closest fit to the configuration $\boldsymbol{X}$, i.e. one that minimises the sum of squared distances of the points to that plane. The remaining PCs are defined recursively in this manner, and individually account for an increasingly smaller amount of the total variance of the p variables of $\boldsymbol{X}$. Thus, the first $q' < q$ principal components give the best linear approximation, and q' can be chosen such that a sufficient proportion of the total variance is preserved, or set to 2 or 3 to facilitate direct visualisation.

Let us define the column centred matrix $\bar{\boldsymbol{X}} = (\boldsymbol{x}_1 - \bar{\boldsymbol{x}}, \ldots, \boldsymbol{x}_N - \bar{\boldsymbol{x}})^T$, with $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$, the sample mean vector. An eigendecomposition of the sample covariance matrix $\boldsymbol{S} = \frac{1}{N-1}\bar{\boldsymbol{X}}^T\bar{\boldsymbol{X}}$ yields eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$, and associated eigenvectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_p$. The $p \times p$ matrix $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{q'}, \boldsymbol{0}, \ldots)$ defines the transformation to the first q' PCs: $\boldsymbol{Y} = \bar{\boldsymbol{X}}\boldsymbol{B}$ [Flury97]. Principal Coordinates Analysis (see Section 3.3.1) on the dissimilarity matrix $\boldsymbol{\Delta} = \left[d_{ij}(\bar{\boldsymbol{X}})\right]$, with d the Euclidean distance (3.1), results in the same configuration $\boldsymbol{Y}$ [Gower66]. However, PCO requires the eigendecomposition of an $N \times N$ matrix of scalar products $\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^T$, whereas in the case of PCA $\boldsymbol{S}$ is only of order $p \times p$. Eigendecomposition of a $k \times k$ matrix is an $O(k^3)$ computation [Press92], and for $p \ll N$, as is normally the case, PCA will be significantly faster. However, PCA is less general because it requires $\boldsymbol{X}$ as input, which can only be constructed for quantitative, ordinal, or binary data, or a combination thereof (see Section 1.2). When PCA can be applied it is equivalent to PCO, and thus the results of Sections 3.5 and 3.6 are applicable to both.

4.4 Comparison

We have gathered 10 large data tables ranging from 2000 to 58000 rows (see Section B.3 in the appendices), and used them as the basis for comparison of incremental MDS and PCA[‖]. Full scaling (see Section 4.2) was performed with the Simulated Annealing algorithm of Section 3.3.6, single scaling with a version of this algorithm modified to position individual objects (rows) with respect to a fixed skeleton. The parameters of the algorithm are identical to that documented in Section 3.4. Each data set was subject to PCA, and incremental MDS with exponent initially set to $\alpha_1 = 2/3$ and then to $\alpha_2 = 3/4$. To avoid degeneracies

[‖]we have based this evaluation on an implementation of PCA by Fionn Murtagh, available from the StatLib Multivariate Archive: http://lib.stat.cmu.edu/multi/pca.c

Table 4.2: Percentage of the total variance explained by the first two or three principal components

data	2D	3D
abalone	94.8	97.6
letter	43.7	56.3
mfeat	91.6	98.7
pageblock	81.4	94.1
sat	84.7	89.9
segment	73.8	83.9
nbody-1	69.1	82.9
nbody-2	70.5	86.9
shuttle	89.3	99.6
spambase	23.6	29.2

of an incremental solution, e.g. a skeleton configuration representing a bad local minimum, the solution was taken as the best in terms of Energy (3.13) of three trials. The target size $\phi = 500$ was used for the initial skeleton (see Section 4.2.2).

Table 4.2 shows the proportion of the total variance explained by selecting only the first two or the first three principal components for each data set, which corresponds to the optimal orthogonal projection from the original high dimensional space to a two- or three-dimensional subspace (see Section 4.3). The third dimension substantially improves on the amount of explained variance over just the first two PCs, except for *abalone* data table, which is already represented well in two dimensions, at 95% of the total variance. On the other hand, *letter* and *spambase* are poorly represented even in three dimensions, and are examples of data tables with relatively uncorrelated attributes, which will pose a challenge to any dimensionality reduction method. However, for other data sets in the test bed over 80% of the total variance is accounted for in three dimensions, resulting in respectable visualisations. Since, PCA is likely to fare much better with three-dimensional visualisations, we decided to consider them also in the statistical comparison.

4.4.1 Statistical Analysis

Our experiment consists of a group of Energy measurements – one for each visualisation method being evaluated – made for every data collection in the test bed, and has been carried out in both two and three dimensions. We chose to perform the statistical analysis with the Friedman test [Siegel56], for the same reasons as outlined in Section 3.5.

Two-dimensional Visualisation

Table 4.3(a) contains Energy measurements in two dimensions for all combinations of a method and a data table. These measurements were subject to the Friedman test with the null hypothesis H_0 of no overall difference in Energy between visualisation methods, and the alternative hypothesis H_1 of at least one method having a different ranking from the others. The values of both Friedman's chi-square statistic and Kendall's coefficient of concordance are not significant at the 5% level, and thus there is no evidence to reject H_0. Consequently, post hoc testing, in the form of the Tukey Multiple Comparison procedure [Keselman77] for example, cannot be carried out.

Nonetheless, certain patterns emerge from Table 4.3(a). Incremental MDS with $\alpha_1 = 2/3$ has the highest rank of Energy in most cases, and it achieves the lowest rank only for the *shuttle* data table. In this case it actually outperforms incremental MDS with $\alpha_2 = 3/4$, which is unusual, but possible with an iterative minimisation technique such as this one. PCA and the α_2 method have an almost identical mean rank, and are one rank apart for most data tables. None of the methods construct a good representation for the *spambase* data set, judging from the Energy values (3.13) alone. This merits a visual inspection, which is presented in Section 4.4.2

Three-dimensional Visualisation

Analogously, we have performed the Friedman test on the Energy measurements for three-dimensional representations, with the same null and alternative hypotheses. As can be seen from Table 4.3(b), this time both Friedman's chi-square statistic and Kendall's coefficient of concordance are significant at the 5% level, allowing H_0 to be rejected in favour of H_1. Tukey Multiple Comparison procedure can now be performed to test for significant differences between method rankings in pairs.

At $p < 0.05$ level the only significant comparison is between incremental MDS with $\alpha_1 = 2/3$ and $\alpha_2 = 3/4$, demonstrating that the latter has a lower overall ranking of Energy measurements. To verify the other two pairwise comparisons, we performed a more powerful one-tailed Wilcoxon signed-rank test [Siegel56]. The test is in agreement with the Tukey Comparison for (PCA, α_2), in that there is no significant difference in their mean ranks. However, PCA is shown to have a significantly lower ranking than α_1 at $p < 0.05$ level.

An informal inspection of Table 4.3(b) leads to the same conclusions as the statistical inference, and also the evaluation of the two-dimensional representations. Incremental MDS with α_1 has the highest rank of Energy for most data tables, and consequently the highest mean rank of all the methods. PCA and the α_2 method achieve a very similar mean rank, and alternate for the lowest and the second lowest rank for individual data sets.

Table 4.3: Friedman test results for Energy

data table	$\alpha_1 = 2/3$	$\alpha_2 = 3/4$	PCA
abalone	0.0210	0.0183	0.0174
letter	0.1413	0.1291	0.1216
mfeat	0.0313	0.0256	0.0578
pageblock	0.0619	0.0509	0.0515
sat	0.0524	0.0418	0.0485
segment	0.0759	0.0617	0.0927
nbody-1	0.2023	0.1360	0.0884
nbody-2	0.1000	0.0779	0.0835
shuttle	0.0192	0.0435	0.0301
spambase	0.3532	0.6638	0.2599
mean rank[a]	2.5	1.7	1.8

statistic	
χ_r^2 [b]	3.8
W [c]	0.19

(a) two dimensions

data table	α_1	α_2	PCA
abalone	0.0090	0.0071	0.0088
letter	0.0686	0.0626	0.0602
mfeat	0.0189	0.0132	0.0169
pageblock	0.0021	0.0014	0.0024
sat	0.0240	0.0206	0.0359
segment	0.0412	0.0269	0.0337
nbody-1	0.1462	0.0729	0.0282
nbody-2	0.0677	0.0243	0.0213
shuttle	0.0191	0.0291	0.0012
spambase	0.9645	0.9540	0.2213
mean rank	2.7	1.6	1.7

statistic	
χ_r^2	7.4
W	0.37

(b) three dimensions

[a] an average of the relative order of Energy for a particular method over all data sets

[b] χ_r^2 is Friedman's chi-square statistic; the critical value at $p = 0.05$ level is 5.99

[c] W is Kendall's coefficient of concordance; it ranges between 0 (no agreement between cases) to 1 (complete agreement); the critical value at $p = 0.05$ level is 0.3

Values in each row are coded in greyscale, based on their relative rank.

Table 4.4: Comparison of running time for PCA and incremental MDS

method	total time[a]	time ratio[b]	linear fit[c]
2D			
PCA	433	n/a	
$\alpha_1 = 2/3$	1155	2.67	2.59
$\alpha_2 = 3/4$	2535	5.86	5.65
3D			
PCA	431	n/a	
α_1	1198	2.78	2.64
α_2	2619	6.08	5.69

[a]average time in minutes required for a single test trial with the test bed
[b]ratio of average time for a given incremental method to that of PCA
[c]the parameter of a least-squares fit of PCA timings to that of each incremental method

Running Time

The total time required for each method to generate visualisations of all data tables in the test bed can be found in the second column of Table 4.4. For incremental MDS each figure is an average over 3 trials; since PCA is an analytical method, only one trial has been performed, and its duration is reported. The total time for each method is very similar across two and three dimensions. With PCA all Principal Components are calculated at once, regardless of how many will actually be used. However, for incremental MDS there is some overhead in calculating distances over an extra dimension, and the total time for three dimensions is slightly higher than for two.

The timings include the calculation of Energy (3.13) for each configuration. For PCA this computation completely dominates the construction of the correlation matrix and its eigendecomposition (see Section 4.3), which itself are only a matter of seconds even for the largest data table. For ease of comparison the ratios of the total time for both variants of the incremental method to that of PCA are given in the third column of Table 4.4. Thus, it can be seen that evaluation of Energy accounts for 37% ($\approx 1/2.7$) of the total time to perform incremental MDS with $\alpha_1 = 2/3$ on the test bed, and 17% ($\approx 1/6$) for $\alpha_2 = 3/4$.

Analogously to Section 3.5.3 we decided to perform a linear regression of PCA timings to that of incremental MDS, to get a more accurate empirical estimate of their ratios. The timings over all data tables in the test bed are collected in a vector $\boldsymbol{m}^{(j,p)}$ for a given method j in p-dimensions. The linear model is expressed as $\boldsymbol{m}^{(i,p)} = a_{i,p}\boldsymbol{m}^{(PCA,p)}$, and the parameter $a_{i,p}$ that gives the best least-squares fit can be interpreted as an estimate of how much slower method i is than PCA in p-dimensions, taking the calculation of Energy into account. The values of $a_{\alpha_1,2}$ and $a_{\alpha_2,2}$ reported in the final column of Table 4.4 agree well with the corresponding

Table 4.5: Statistics for two-dimensional visualisations of *mfeat*, *segment*, and *spambase* data tables

	Energy			
data	PCA	$\alpha_1 = 2/3$	$\alpha_2 = 3/4$	MDS
mfeat	0.0578	0.0313	0.0256	0.0201
segment	0.0927	0.0759	0.0617	0.0469
spambase	0.2599	0.3532	0.6638	0.1460
		Stress		
mfeat	0.0127	0.0128	0.0108	0.0141
segment	0.0441	0.0418	0.0310	0.0261
spambase	0.3120	0.0955	0.0861	0.1401
		time		
mfeat	14	123	235	2376
segment	43	190	334	3582
spambase	495	960	1535	8228

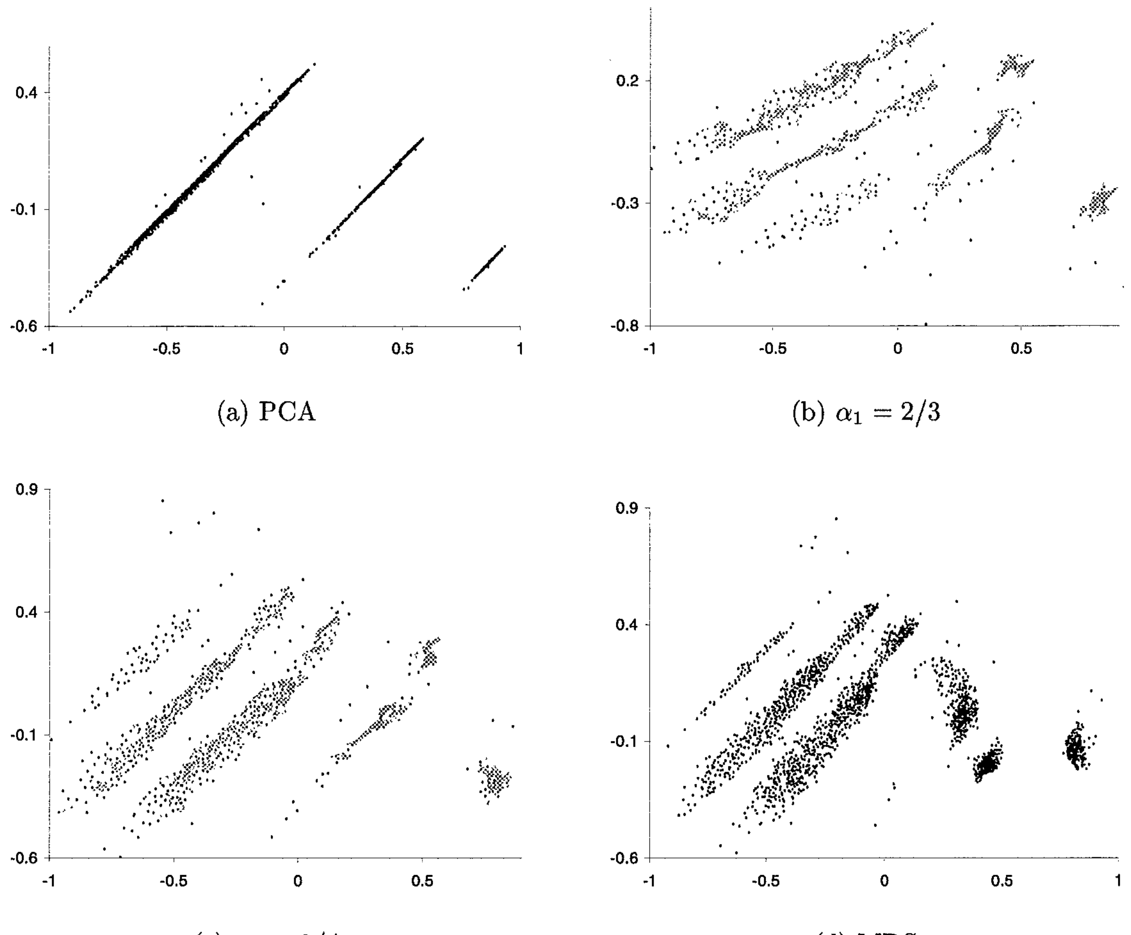
running time ratios, and even better with $a_{\alpha_1,3}$ and $a_{\alpha_2,3}$ respectively, signifying that the estimation is accurate.

The time to calculate Energy of a configuration is comparable to that of a single iteration of full scaling (see Section 4.2) with this configuration, since both computations involve the inspection of the entire dissimilarity matrix $\mathbf{\Delta}$. Therefore, following an incremental MDS procedure with full scaling of the complete configuration will add substantially to the running time, for each additional iteration.

4.4.2 Qualitative Evaluation

We have chosen the three smallest data tables to present visually here. They are easier to represent in a limited space, to allow a side-by-side comparison, but manage to bring out key differences between the methods, nonetheless. We have been able to perform standard MDS on these data due to their manageable size, and include it in the evaluation of PCA and incremental MDS, as the limiting case of the latter. For ease of comparison the relevant Energy values (3.13) of Table 4.3(a) are repeated alongside the results for standard MDS in Table 4.5; Stress (3.11) and the running time for each data set are also given. In all cases MDS achieves the lowest Energy, however, at a marked increase in the running time. Stress results are less consistent, which is understandable as it is Energy that has actually been minimised; thus observations of Section 4.2.3 are replicated.

Visualisations produced by all the methods are arranged side by side in Figures 4.2–4.4 for each data set respectively. Analogously to Section 3.6, MDS con-



(a) PCA

(b) $\alpha_1 = 2/3$

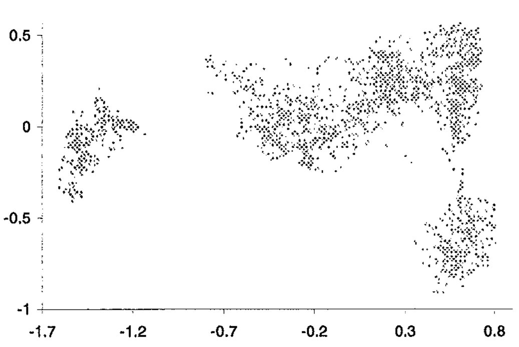
(c) $\alpha_2 = 3/4$

(d) MDS

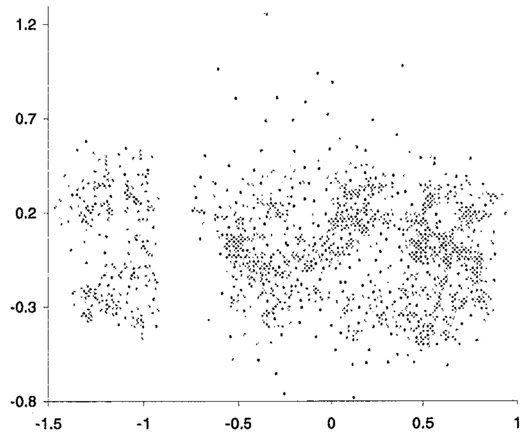
Figure 4.2: Visualisations of the *mfeat* data set. Black dots in the incremental MDS plots denote the skeleton configuration

figurations have been transformed by Procrustes Analysis [Borg97, Cox94] to have the same orientation as the corresponding PCA configuration. Tucker's congruence coefficient (3.28) has been used to ascertain the agreement between configurations in pairs [Tucker51], and all possible combinations are given in Table 4.6.

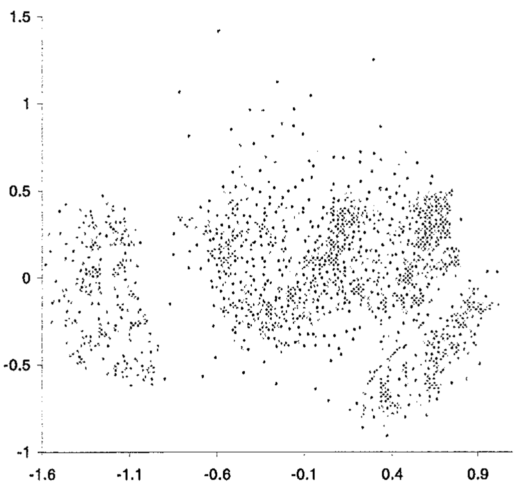
The overall structure of the *mfeat* data set is represented well by the first two principal components. Three distinct clusters can be seen in Figure 4.2(a); however, the intra-cluster detail is not discernible, and this where the 8% of unexplained variance (see Table 4.2) must lie. Incremental and standard MDS plots in Figures 4.2(b)–4.2(d) are superior, in that they spread the cluster contents out, and portray more clusters than PCA. $\alpha_2 = 3/4$ and MDS plots are very similar, which is confirmed by the highest value of the Tucker's congruence coefficient (3.28) for this pair, reported in Table 4.6(a). Note that the position of marker points is in a good agreement for these two configurations, and thus attests the validity of the incremental MDS procedure. The match between $\alpha_1 = 2/3$ and



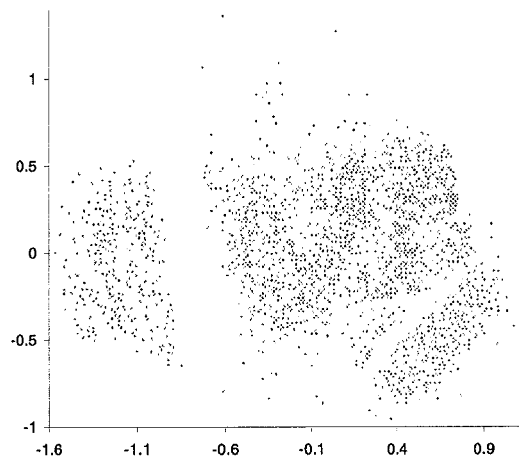
(a) PCA



(b) $\alpha_1 = 2/3$



(c) $\alpha_2 = 3/4$

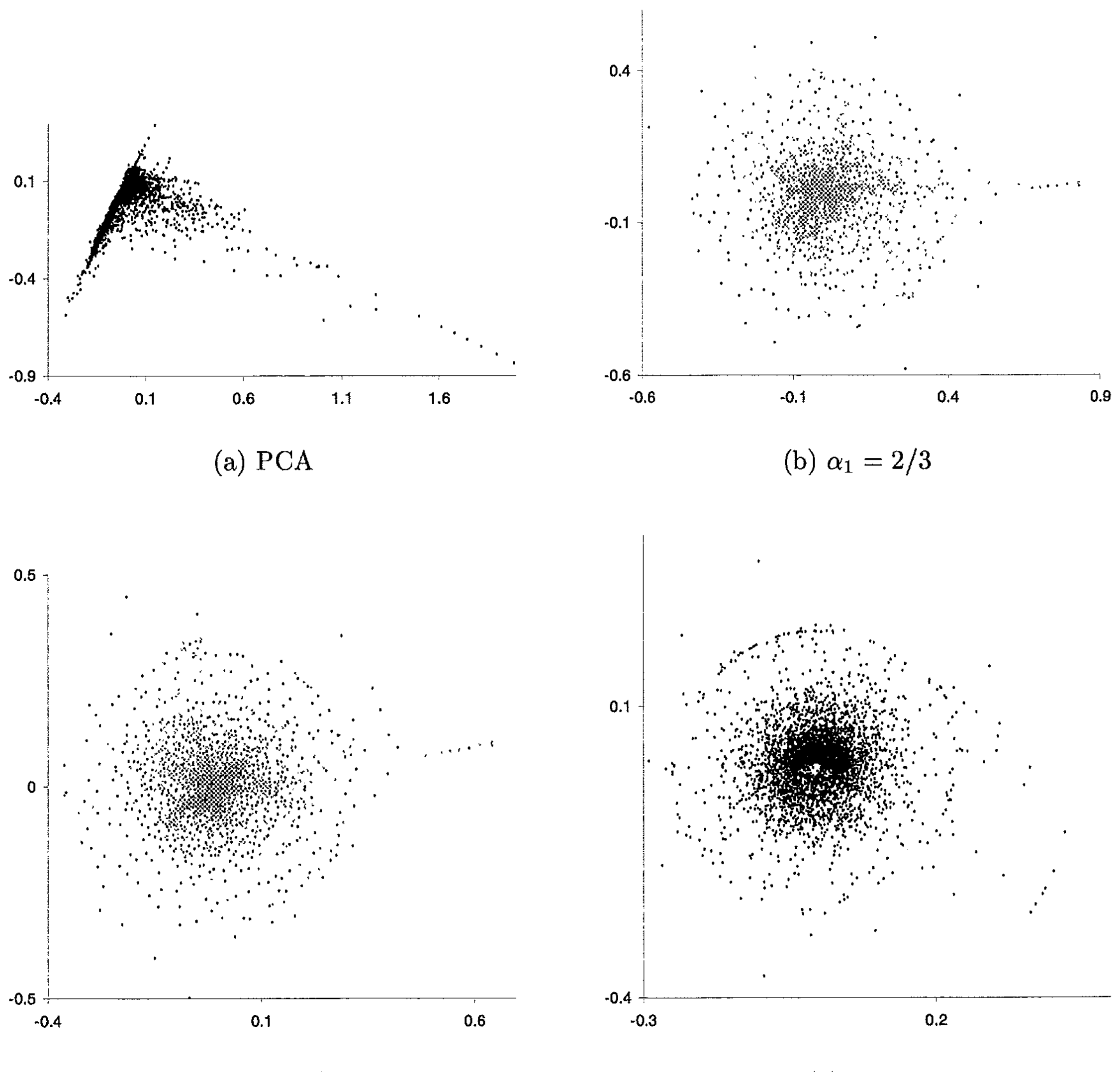


(d) MDS

Figure 4.3: Visualisations of the *segment* data set. Dark dots in the incremental MDS plots denote the skeleton configuration

MDS configurations is not as good, due to a smaller number of marker points defining the skeleton configuration: $n_1^{(\alpha_1)} = 159$ versus $n_1^{(\alpha_2)} = 299$.

It is possible to identify three clusters in the PCA representation of the *segment* data set in Figure 4.3(a). There are some dense clumps of points, highlighting areas of unexplained variance (see Table 4.2). Figures 4.3(b), 4.3(c), and 4.3(d) in turn exhibit an increasingly better dispersion of points, showcasing the bias of Energy (3.13) towards preserving local detail (see Section 3.2.4 for more details). Again, α_2 and MDS plots are the most similar visually, and objectively according to the congruence coefficient (see Table 4.6(b)). The distinction between two of the clusters is lost in the α_1 configuration, suggesting that an insufficient size of the skeleton has been specified.



(a) PCA (b) $\alpha_1 = 2/3$

(c) $\alpha_2 = 3/4$ (d) MDS

Figure 4.4: Visualisations of the *spambase* data set. Black dots in the incremental MDS plots denote the skeleton configuration

The *spambase* data set presents a challenge to PCA, as according to Table 4.2 only 24% of the total variance is explained by the first two PCs. Only a few outlying data points are represented well in Figure 4.4(a), the remaining ones are grouped in a single clump, with little apparent structure. Incremental MDS is able to spread the contents of the clump, and a dense distribution of the points can be seen in the middle of Figures 4.4(b) and 4.4(c), surrounded by layers of other points, mainly markers. Such a concentric arrangement is indicative of a severe mismatch between dimensionality of the representation and the original data [deLeeuw84]. The two incremental plots are similar, and the congruence coefficient, reported in Table 4.6(c), is much higher for this pair than any other. As can be seen from Figure 4.4(d), standard MDS pushes more points towards the

Table 4.6: Congruence coefficient for pairs of configurations

	PCA	$\alpha_1 = 2/3$	$\alpha_2 = 3/4$
α_1	0.9915		
α_2	0.9884	0.9849	
MDS	0.9867	0.9810	0.9928

(a) *mfeat*

	PCA	α_1	α_2
α_1	0.9751		
α_2	0.9835	0.9833	
MDS	0.9800	0.9796	0.9920

(b) *segment*

	PCA	α_1	α_2
α_1	0.8280		
α_2	0.8209	0.9762	
MDS	0.7441	0.8880	0.8940

(c) *spambase*

perimeter, and achieves a much lower value of Energy (see Table 4.5); however, it cannot bridge the dimensionality gap, and visually the effect is not very different to incremental MDS plots.

For the *spambase* data set the subjective observations are in contrast to the statistical analysis of Section 4.4.1. Incremental MDS is judged to be much worse than PCA by the Energy function (3.13), as can be gleaned from Table 4.5. It seems that local detail cannot be represented well for this data set, unless all dissimilarities are considered, as is the case for standard MDS. However, the converse ranking of PCA and incremental MDS is true for Stress (3.11). This loss function assesses the absolute error of a configuration, and thus agrees with the overall, subjective impressions. The reader is referred to Section 3.2.4 for a further comparison of Energy and Stress.

4.5 Discussion

The incremental Multidimensional Scaling method presented here alleviates the high time and space complexity associated with standard MDS. This method is capable of visualising data collections in excess of 100,000 objects at moderate cost.

Inevitably, some additional inaccuracy is introduced into configurations. However, structurally significant objects will be represented accurately with respect to one another, and error will be spread over a large number of less significant objects. If enough computing resources are available the resulting configuration can be refined by submitting it as a starting point for a least-squares MDS procedure.

Multivariate data tables can alternatively be visualised with Principal Components Analysis, provided that none of the attributes are measured on a nominal scale. For such data PCA is equivalent to Principal Coordinates Analysis, at a fraction of its computational cost. As has been pointed out in Section 3.7, these methods will not represent the local proximity of objects accurately. However, incremental MDS also compromises local detail to achieve its speed up, and instead focuses on preserving the overall shape of the configuration. The difference is that the trade-off between quality and algorithm responsiveness can be varied, and approach the ultimate quality of least-squares MDS in the limiting case, with the associated long running time.

The objective evaluation of incremental MDS versus PCA was somewhat inconclusive, because of the limited number of data collections on which it was based. Real-world data of the required size is difficult to obtain, but more importantly an exhaustive assessment of the methods with each data set is very time consuming. The key observation is that at the trade-off level where incremental MDS matches PCA for overall quality, the former is much more expensive computationally. However, visual inspection of configurations produced by both methods reveals that objective measurements are misleading in this case, and that incremental MDS provides more pleasing arrangements than PCA, even at a lower objective quality and quicker response setting.

4.6 Related Work

A least-squares MDS algorithm has been proposed that computes the configuration update in linear time in the number N of objects [Chalmers96]. When calculating a new position of an object only two small subsets of the remaining objects are considered. One of them contains the most similar objects to the given object; the other is sampled at random. By keeping the sizes of these subsets constant an $O(N)$ time complexity is achieved for a single iteration. Assuming that $O(N)$ iterations are necessary to find an optimal configuration, the algorithm can be seen to have $O(N^2)$ overall time complexity. There are two possible disadvantages to this approach: instability due to optimising against a changing set of objects, and inaccuracies in representing large dissimilarities, which can result in a distorted configuration.

FastMap [Faloutso95] is an analytical MDS algorithm that achieves $O(kN)$ running time, where k is the number of dimensions in the configuration. It is an enormous improvement over PCO (see Section 3.3.1), which is expensive as it involves the eigendecomposition of a $N \times N$ matrix – an $O(N^3)$ computation [Press92]. A line passing through the two most distant (dissimilar) data points in the multidimensional space approximates the first principal axis (principal component), which is a line that yields the maximum variance when data points are orthogonally projected onto it (see Section 4.3). All data points are projected onto the hyperplane perpendicular to this line. Successive applications of this procedure to the resulting subspace approximate the remaining principal axes. The resulting axes are necessarily orthogonal. Although very fast, this method only approximates a PCO solution, which itself is likely to be inferior to a least-squares MDS configuration (see Sections 3.5 and 3.6).

An incremental arrangement method [Cohen97] has been proposed to speed up drawing of undirected graphs, and avoid suboptimal solutions. The order of N vertices to be introduced into the drawing is determined by performing a depth-first search from an arbitrary vertex, and reshuffling this list so that every $\frac{N}{i}$th vertex is taken first, then every $\frac{N}{i^2}$th, and so on, ignoring duplicates. The first batch of vertices is placed using standard MDS. The next batch is then added to the partial drawing, by positioning each new vertex in proximity of the two vertices introduced previously that are its nearest neighbours in the graph. Another round of MDS follows, and these two stages are applied alternately while there are vertices to be added. The algorithm has been shown empirically to achieve a factor of 2 to 10 speed improvement.

Chapter 5

Proximity Grid

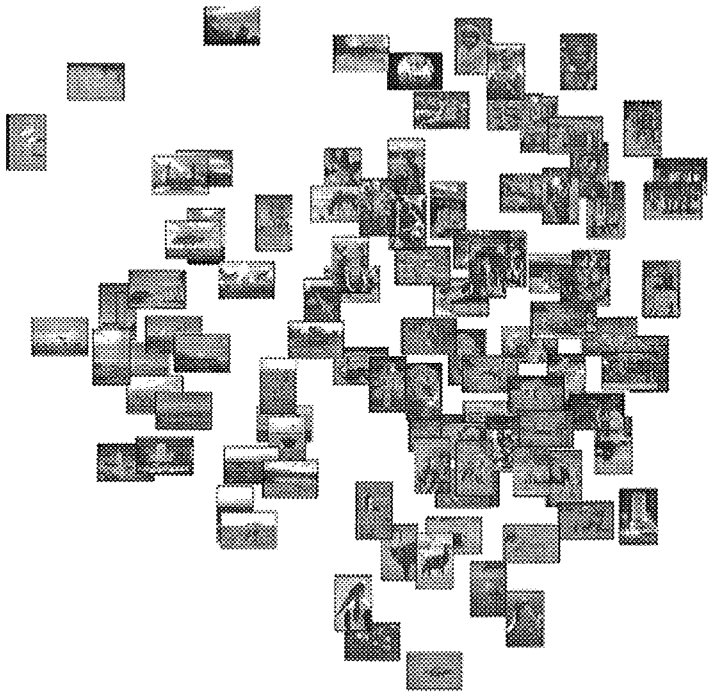
This chapter is concerned with Proximity Grid – a visualisation technique that allows information to be presented with high density. Each element of an abstract data collection is represented within a single cell of the grid, and thus considerable detail can be shown without overlap. The proximity relationships are preserved by clustering similar elements in the grid, and keeping dissimilar ones apart. Proximity Grid is thus a counterpart of Multidimensional Scaling in a discrete domain. Four algorithmic solutions to this problem are presented, and evaluated in terms of output quality and running time. A recommendation for algorithm selection is made based on the trade-off between these criteria.

5.1 Origins of the Problem

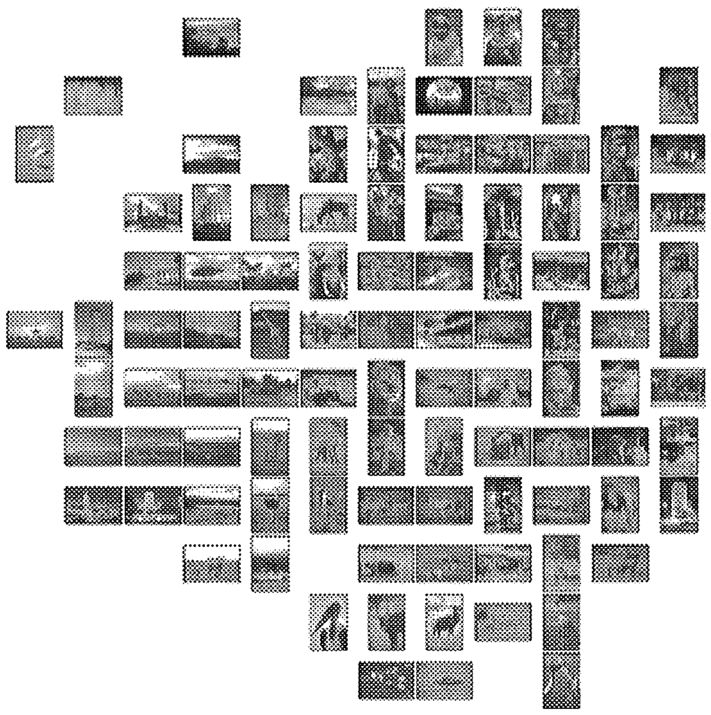
An experimental evaluation of MDS arrangements versus random grid arrangements, in the context of browsing a collection of images, demonstrated the superiority of the former for a basic search task [Rodden99a]. However, in a post-experimental questionnaire subjects reported overlap of image thumbnails in MDS arrangements to cause them difficulty in finding a target image. Some actually preferred the regularity of a random grid, because it made scanning the collection easier. An arrangement method that combined the best features of both configuration types – here referred to as *Proximity Grid* – was proposed.

Because the MDS analysis does not take the size of configuration elements into account, its direct use for visualisation may lead to partial or complete overlap of icons representing the objects from a collection. Figure 5.1(a) is an MDS visualisation of an image collection, with many thumbnails having their detail obscured. By forcing icon centres to lie on a grid, a minimum amount of separation can be ensured, while still keeping similar icons (objects) together. The effect of transforming the MDS configuration to a proximity grid can be seen in Figure 5.1(b).

The problem of overlap is less serious in three dimensions. The ability to view a 3D visualisation from different viewpoints aids in separating the icons, and thus alleviates the problem to a large extent. Also, the size of icons can be reduced to limit overlap, and compensated by the natural use of zooming and panning. These



(a) MDS arrangement generated by the KYST algorithm ($\sigma = 0.0692$)



(b) 12×12 proximity grid generated by the SWO algorithm ($\sigma = 0.0789$)

Figure 5.1: 100 images of Kenya arranged by visual similarity. These images have been taken from the Corel stock photograph collection

counter measures are not suitable for a flat representation, whose primary use is to provide an informative overview of a data collection. Therefore, we restrict our attention to two-dimensional grids, but note that the theory and algorithms can easily be extended to higher dimensionality.

5.2 Algorithms

The purpose of the algorithms described below is to position a set of n objects in a grid, such that similar objects occupy neighbouring cells, and are separated from dissimilar objects, where the dissimilarities between all pairs of objects are collected in matrix $\boldsymbol{\Delta}$, as in Chapter 3. The size of the grid, i.e. the number of cells, can be greater than the number of objects. As the number of spare cells rises, there is more freedom in placing the objects, and a more faithful representation of pairwise similarity can be achieved. In the limiting case the grid will become so sparse that it will be equivalent to a continuous MDS configuration, especially when taking the discrete nature of computer displays into account. Therefore, the accuracy of a grid algorithm at a certain level of grid density can be assessed against the corresponding continuous configuration.

The quality of a proximity grid $\boldsymbol{Y} \in \mathbb{N}^2$, where $\mathbb{N}$ denotes the set of natural numbers, can be established by evaluating one of the loss functions discussed in Section 3.2 for a given $\boldsymbol{Y}$ and $\boldsymbol{\Delta}$. An algorithm for generating proximity grids is essentially a numerical minimisation procedure for such a loss function. As a consequence of specifying this minimisation problem over a discrete domain – $\mathbb{N}^2$ – the choice of a heuristic is limited, as any method relying on derivatives cannot be applied.

When using Stress (3.11) small dissimilarities will not be modelled faithfully, because even a large relative error will have negligible impact on the value of (3.11); see Section 3.2.4 for a discussion and comparison with Energy (3.13). This property is actually an advantage in the case of a discrete (grid) configuration, because similar objects cannot be closer than the cell separation, and will not be penalised for it. Consequently, we have chosen Stress as the basis of our grid algorithms. KYST[†] is a publicly available MDS program that works with Stress-1 (3.9) [Kruskal78b], which is equivalent to Stress after a suitable scaling transformation (see Section 3.2.3 for details). Hence, we used KYST for generating continuous MDS configurations corresponding to proximity grids.

5.2.1 Greedy

The idea behind this algorithm is to start with a continuous MDS configuration, and discretise it so that object locations are snapped to the grid. Naturally, it is possible for two objects to be mapped to the same grid cell, and many strategies could be employed to resolve this. Since we wanted to avoid introducing another

[†]KYST version 2a is available from Netlib (www.netlib.org)

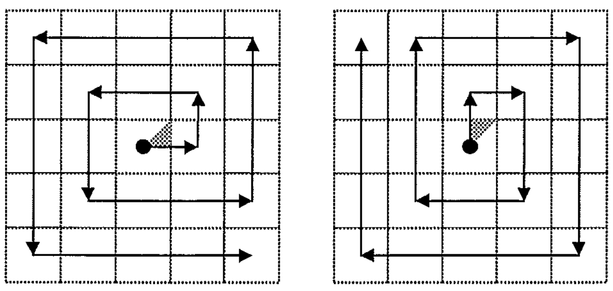


Figure 5.2: Example of spiral search in a grid. The grey triangle represents the set of all possible positions of point y_i that would result in this particular orientation of the spiral. The black circle denotes the optimal cell c for this point

numerical optimisation process, we adopted a greedy approach. This will only give an approximate solution, but the computational cost is low, and the amount of introduced error can be controlled by adjusting the grid size.

The location y_i of each object i in the configuration is considered in turn. If the closest grid cell c to y_i is unoccupied, i is allocated to it. Otherwise, a close by empty cell e is found by performing a spiral search starting from cell c. The second cell visited by the square spiral is the second closest cell to y_i, taking the Euclidean distance into account. This determines the orientation of the spiral in an unbiased way. Figure 5.2 is an example of a spiral search when c is below and to the left of y_i; the remaining three cases are analogous. If there are at least as many grid cells as objects this procedure is guaranteed to find e, though this might not be the closest empty cell to y_i.

Once e is found three different strategies can be adopted, as illustrated in Figure 5.3:

1. *empty* – i is allocated to e
2. *swap* – the object allocated to c is relocated to e, and i is allocated to c
3. *bump* – objects allocated to cells on the line from c to e are relocated outwards by one cell, and i is allocated to c. These cells can be determined by applying the line drawing algorithm of computer graphics.

Note that a *bump* is equivalent to a *swap* when e is adjacent to c.

The order in which objects are considered is important, and should be independent of permutations of the input configuration. We define a complete weighted graph with a vertex for each object in the collection, and take each edge weight to be the dissimilarity between the corresponding pair of objects. The Minimum Spanning Tree (MST) calculated for this graph then defines such a unique ordering of objects (see Section 4.2.1). The shortest MST edges link the most closely related objects. Considering them first in the *empty* strategy will ensure that they occupy neighbouring cells. The same effect will be achieved by considering them last in the *swap* and *bump* strategies.

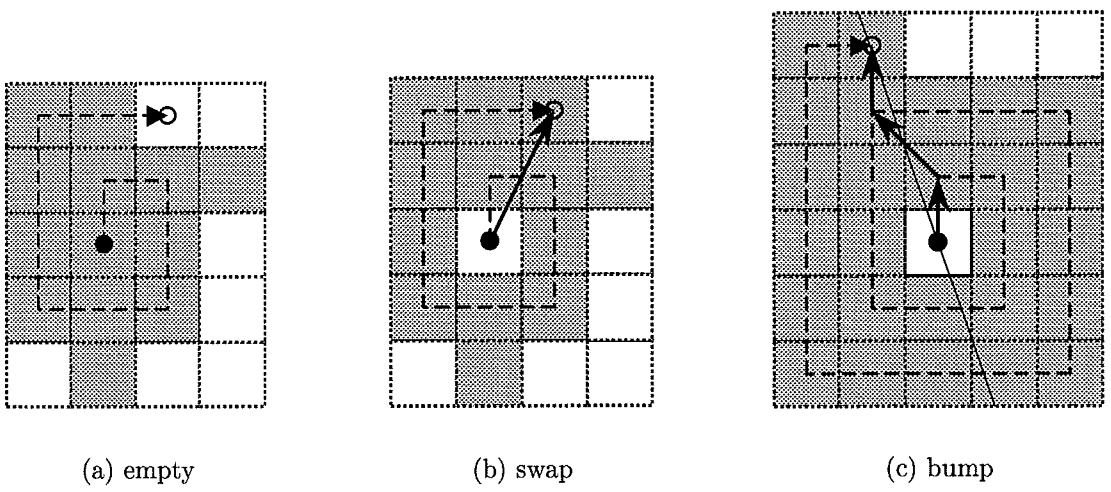


Figure 5.3: Examples of different grid allocation strategies. The solid black circle denotes the optimal cell, and the empty circle shows the empty cell found during the spiral search

To identify the best strategy we have performed a rigorous evaluation, which is presented in Section 5.3.1. The *bump* strategy is significantly better in terms of Stress (3.11) for proximity grids that it generates. This can be attributed to its tendency of spreading error (3.3) over many objects, rather than positioning some optimally, and delegating potentially large error to others, as is the case for the other strategies. Henceforth, we refer to the greedy algorithm with the *bump* strategy as *Greedy1.*

5.2.2 Improved Greedy

One obvious area of improvement to *Greedy1* is to replace the spiral search by an exact procedure to find the closest empty cell to the original object location y_i. A brute force approach is to calculate the distance between y_i and the centre of every cell in the grid, sort these distances in the ascending order, and pick the first corresponding cell that is empty. The same effect can be achieved by evaluating the distances lazily, by considering increasingly larger sub-grids around c. We start with the eight cells that immediately surround c, and enter them into a heap [Press92], with the heap condition based on their distance to y_i, and the closest remaining cell to y_i as the root. At each step the root cell r is examined, if it is empty the search is terminated; otherwise, it is removed from the heap. If r falls on the boundary of the current sub-grid, the sub-grid is enlarged in this direction by entering a new layer of cells into the heap. The heap condition is subsequently restored. We refer to this modified grid algorithm as *Greedy2.*

5.2.3 "Squeaky Wheel" Optimization

Section 5.2.1 highlighted that the order in which objects are considered for inclusion in the grid by the greedy algorithm is important. In *Greedy1* and *Greedy2* we decided to derive this order from the data themselves, so that closely related objects would be given preferential treatment. In general, this will not give the best arrangement in terms of Stress (3.11), and there will be other orderings that will give better results. One could envisage a procedure that will continually permute the order of objects, run the greedy algorithm with it, and evaluate Stress in search of the minimum. An obvious candidate is a Genetic Algorithm [Goldberg89] (also see Section 3.3.4) or Simulated Annealing [Dowsland95] (also see Section 3.3.6). However, a much better use for these techniques is to search directly for the best allocation of objects to grid cells, as in Section 5.2.4, bypassing the greedy and continuous MDS steps.

"Squeaky Wheel" Optimization (SWO) [Joslin99] is an effective technique for coupling a greedy algorithm with a reordering (reprioritisation) scheme. Once a solution is constructed, it can be evaluated to find objects that contribute the most to the value of the loss function, and these objects can be promoted in the sequence, so that they will be handled earlier and probably better on the subsequent greedy run. This construct/analyse/prioritise cycle continues for a set number of iterations, or until an acceptable solution is found. Typically, the sequence evolves so that easy to deal with objects sink to the back, whereas troublesome ones stay at the front.

In our SWO algorithm the construction phase is essentially the entire *Greedy2*, with the *empty* allocation strategy (see Section 5.2.1). The SWO algorithm will automatically achieve the effect of spreading the representation error (3.3), and an explicit mechanism in the form of the *bump* strategy will actually interfere with it. The initial sequence is the ascending MST order, which is the appropriate choice for the *empty* strategy (see Section 5.2.1).

Every object is subsequently analysed, and the distance from the centre of its grid cell c to the original continuous location y_i constitutes its *blame factor*. If c is the closest cell to y_i the blame factor is set to 0. The prioritisation phase consists of a partial bubble sort. On a single pass from the front of the sequence to the back, every object that has a positive blame factor is swapped with its predecessor, and its blame is reduced by 1. The net effect is that objects move forward in the sequence proportionally to their blame. The SWO cycle is repeated a fixed number of times ($SWO.iterationlimit$ parameter), and the grid configuration with the lowest value of Stress (3.11) constitutes the solution.

5.2.4 Genetic Algorithm

For a brief introduction to Genetic Algorithms (GAs) the reader is referred to Section 3.3.4. It has been relatively straightforward to adapt the GA for continuous MDS to a Proximity Grid one, which is a clear demonstration of how generic this heuristic strategy is. The details of the selection scheme and that of forming a new generation of the chromosome population remain unchanged, and are given in the final three paragraphs of Section 3.3.4. The only exception is that the fitness of a chromosome is based on Stress (3.11) instead of Energy (3.13). The low-level issues of selecting the coding of the problem and associated crossover and mutation operators have to be reconsidered, and are discussed below.

An allocation of n objects to k grid cells can be compactly represented by a permutation of n cell identifiers, taken from an alphabet of cardinality k. Thus, a *non-binary coding* seems to be the most appropriate for this problem [Goldberg89]. A solution is encoded as a vector $\boldsymbol{v}$ of n cardinal numbers, where object i is allocated to cell v_i, determined by counting cells sequentially from top left to bottom right. A pair of solutions $\boldsymbol{v}^{(p_1)}$ and $\boldsymbol{v}^{(p_2)}$ is recombined by applying the *cycle crossover* (CX) operator [Goldberg89], to yield a new pair of chromosomes $\boldsymbol{v}^{(c_1)}$ and $\boldsymbol{v}^{(c_2)}$. This operator ensures that the value $v_i^{(c_1)}$ of each child gene i comes from the first parent $v_i^{(p_1)}$ or the other $v_i^{(p_2)}$, and that the whole solution is feasible, i.e. no two elements of $\boldsymbol{v}^{(c_1)}$ are the same; $\boldsymbol{v}^{(c_2)}$ is simply the complement of $\boldsymbol{v}^{(c_1)}$. Inheriting a gene from a particular parent might require another gene to be taken from the same parent, and so on, in order to meet the above conditions. This dependency is circular, and CX operates by copying each cycle from a randomly selected parent.

The most appropriate form of mutation to use here is *exchange mutation* [Reeves95]. An object i is drawn at random from the range $[1, n]$, and a new cell a is randomly selected for it from the range $[1, k]$. If there exists j such that $v_j = a$, i.e. cell a is already occupied, v_j and v_i exchange their values; otherwise v_i simply becomes a. As argued in Section 3.3.4, the relative importance of crossover and mutation changes throughout the optimisation process, and an adaptive rate of mutation is likely to work best. Thus, whenever CX produces a pair of chromosomes that is identical to the input pair, exchange mutation is performed on both chromosomes. This form of mutation reintroduces diversity only when needed, and does not interfere with crossover otherwise.

5.3 Comparison

To evaluate the performance of each algorithm, we used a test bed consisting of 53 data collections: 2 dissimilarity matrices, 2 image collections, 6 graphs, and 43 data tables (see Section B.1 in the appendices for details). The number of objects in a collection ranged from 9 to 506, with the median value of 107. For each collection we applied *KYST* to calculate a continuous MDS configuration, representing a minimum of Stress (3.11). Since *KYST* is based on the steepest descent technique [Press92], and therefore can only be expected to find a local minimum, we decided to run it 10 times from a random initial configuration, and only use the final configuration with least Stress. These continuous configurations served as input to *Greedy1*, *Greedy2*, and *SWO* algorithms. Subsequently, these three algorithms and *GA* were set to generate proximity grids with varying levels of density for each data collection. Grids from 10 to 100% full at the 10% density increments were used, and the resulting Stress recorded for each. We only considered square grids, and so the size was rounded up to the nearest square number. For a small data collection this could result in identical grid sizes for different density levels, and we avoided duplicating the experiment in these cases.

Parameters of the algorithms were fixed to the following values:

- $SWO.iterlimit = GA.iterationlimit = KYST.iterationlimit = 1000$
- $GA.populationsize = 32$, see Section 3.4 for the justification.

5.3.1 Statistical Analysis

Our experiment consists of a group of related measurements, made for every data collection in the test bed. We decided to rely on the Friedman test [Siegel56] for the analysis, for the same reasons as given in Section 3.5. Before proceeding with the main experiment we had to establish which allocation strategy (see Section 5.2.1) is best, and we present our findings in the first subsection. The remaining subsections are concerned with the main experiment. Each experimental group consists of 40 measurements of Stress (3.11), one for each combination of grid density level and algorithm. There are three possible ways of grouping these measurements and structuring the analysis: by algorithm, by grid density, and complete.

Allocation Strategy Analysis

To determine whether *empty*, *swap*, or *bump* allocation strategy (see Section 5.2.1) is best to use with the greedy algorithm, we have implemented all three, and tested with the test bed of data collections. The denser the grid the more allocation clashes will occur, and require the intervention of a particular allocation strategy. Thus, we focused on the densest square grids possible: $m \times m$, where $m = \lceil \sqrt{n} \rceil$, and n is the number of objects to be positioned in the proximity grid.

Table 5.1 reports the results of the Friedman test with the null hypothesis H_0 of no difference between strategies with regard to Stress (3.11), and the alternative hypothesis H_1 of at least one strategy having a different ranking from the others. The values of both Friedman's chi-square statistic and Kendall's coefficient of concordance are statistically significant at a level exceeding 0.001, which allows H_0 be rejected in favour of H_1. Having done so, we can apply the Tukey Multiple Comparison procedure to test for significant differences between strategy rankings in pairs [Keselman77].

As can be gleaned from Table 5.2, all Tukey comparisons are significant at the 5% level, implying that there are definite differences between strategies, and allowing a complete ordering to be derived: *bump* has the lowest rank, meaning that it consistently achieves the lowest Stress, *empty* comes second, and *swap* performs the worst. Non-parametric tests will not establish what the actual differences are, however. A linear regression of *bump* measurements $\boldsymbol{m}^{(bump)}$ to ones for *empty* and *swap* in turn can provide an answer: $\boldsymbol{m}^{(i)} \approx a_i \boldsymbol{m}^{(bump)}$, where $\boldsymbol{m}^{(j)}$ is the

Table 5.1: Friedman test results for allocation strategies

strategy	rank sum[a]	mean rank[b]
empty	102.0	1.93
swap	147.5	2.78
bump	68.5	1.29

statistic	
χ_r^2 [c]	60.76
W [d]	0.573

[a]rank sum is the sum of the relative order in terms of Stress for a particular strategy over all 53 cases

[b]mean rank is the corresponding rank sum divided by the number of cases

[c]χ_r^2 is Friedman's chi-square statistic; the critical value at $p = 0.001$ level is 13.82

[d]W is Kendall's coefficient of concordance; it ranges between 0 (no agreement between cases) to 1 (complete agreement); the critical value at $p = 0.001$ level is 0.13

Table 5.2: Tukey Multiple Comparison results for allocation strategies

rank comparison	difference[a]	Q [b]
swap>bump	79.0	10.85
swap>empty	45.5	6.25
empty>bump	33.5	4.60

[a]the difference in rank sums; see Table 5.1

[b]Q is Tukey's statistic for multiple comparisons; the critical value at $p = 0.05$ level is 3.31

vector of Stress measurements for all 53 data collections under condition j. The best least-squares fit is for $a_{empty} = 1.05$ and $a_{swap} = 1.23$, meaning that *empty* is 5% worse than *bump*, and *swap* is 23% worse, when considering 100% dense grids. These differences will gradually diminish as the grids get sparser, and disappear completely, when no allocation clashes occur, and there is no need to employ one of these strategies. Nonetheless, *bump* appears to be the best strategy, and we have based *Greedy1* and *Greedy2* on it.

Algorithm Analysis

There are 53 groups, one for each data collection, of 4 measurements of Stress (3.11) for the *Greedy1*, *Greedy2*, *SWO*, and *GA* algorithm respectively. Such a block of measurements is made for a particular level of grid density, and we analyse them in the descending order, starting with grids that are 100% full. The second row of Table 5.3 reports the results of the Friedman test with the null hypothesis H_0 of no difference between algorithms with regard to Stress at

Table 5.3: Friedman test results for the algorithm grouping

density	Greedy1	Greedy2	SWO	GA	χ_r^2 [a]	W [b]
100%	3.57	3.42	2.00	1.02	141.55	0.890
90%	3.63	3.35	2.00	1.02	142.74	0.898
80%	3.70	3.28	2.00	1.02	144.49	0.909
70%	3.59	3.33	2.00	1.08	134.77	0.848
60%	3.66	3.28	1.92	1.13	134.90	0.848
50%	3.63	3.20	1.94	1.23	120.18	0.756
40%	3.72	3.13	1.92	1.23	125.71	0.791
30%	3.69	3.14	1.87	1.30	119.76	0.753
20%	3.48	2.82	1.81	1.89	65.76	0.414
10%	3.13	2.69	1.78	2.40	34.06	0.214

[a]χ_r^2 is Friedman's chi-square statistic; the critical value at $p = 0.001$ level is 16.27
[b]W is Kendall's coefficient of concordance; the critical value at $p = 0.001$ level is 0.102
The figures in columns 2–5 are mean ranks across 53 observations

100% grid density, and the alternative hypothesis H_1 of at least one algorithm having a different ranking from the others. The values of both Friedman's chi-square statistic and Kendall's coefficient of concordance are statistically significant at a level exceeding 0.001, which allows H_0 be rejected in favour of H_1. The Tukey Multiple Comparison procedure can now be performed to test for significant differences between algorithm rankings in pairs.

All pairwise comparisons, except for the one involving *Greedy1* and *Greedy2*, are significant at the $p < 0.05$ level, meaning that *GA* achieves the best performance at 100% grid density, followed by *SWO*, and the greedy algorithms are tied in last place. We decided to perform a one-tailed Wilcoxon signed-rank test, because it is considerably more powerful [Siegel56], to verify the tie of the greedy algorithms. It produced a p value of 0.055, which just misses our nominal 5% limit, and supports the Tukey comparison.

The analysis of the remaining grid densities is analogous. The results of the Friedman test in every case allow H_0 be rejected in favour of H_1 at the 0.001 significance level. The outcome of the Multiple Tukey comparisons is essentially the same for densities between 100 and 40%, i.e. *GA* always comes first, followed by *SWO*, and the greedy algorithms jointly in last place. However, the Wilcoxon test for each of the greedy pairs is significant at the 5% level, except for 70% density with $p = 0.06$. Thus overall, there is a difference between the greedy algorithms, with *Greedy2* performing significantly better.

For 30 and 20% density Tukey comparisons between *GA* and *SWO* are no longer significant at the 5% level. The comparison between *Greedy1* and *Greedy2* at 30% density is not significant either. However, the corresponding Wilcoxon tests are significant at the 5% level, and so the same ordering of the algorithms

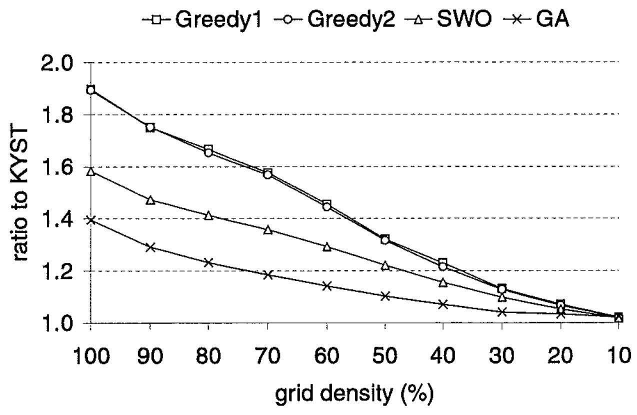


Figure 5.4: Linear regression comparison of grid algorithms. The closer a combination of algorithm and grid density is to 1 (KYST) the better

holds: *GA* first, then *SWO*, *Greedy2*, and *Greedy1* last.

The situation is less clear at 10% density. The only significant Tukey comparisons are: *Greedy1-SWO*, *Greedy1-GA*, *Greedy2-SWO*. Additionally, the Wilcoxon test is significant at the 5% level for *Greedy1-Greedy2*, but is not significant for *GA-SWO*; the test for *Greedy2-GA* produces a p value of 0.061. If the last result is accepted, the following ordering can be inferred: *SWO* and *GA* are tied in first place, followed by *Greedy2*, and *Greedy1* is last.

To get a sense of the actual differences in performance between algorithms, we have fitted a linear regression to the measurements for each combination of algorithm and density. The values of Stress (3.11) for *KYST*, which can be seen to correspond to grid density of 0, thus serving as a suitable reference, were used as the model predictor:

$$m^{(algorithm,density)} \approx a_{algorithm,density} m^{(KYST)}$$

where $m^{(i)}$ is the vector of Stress measurements for all 53 data collections under condition i. The value of a_i that gives the best least-squares fit of the model is plotted in Figure 5.4 for each level of density, with separate series for each algorithm. The further a_i is from 1 the greater the difference of the corresponding Stress measurements between condition i and *KYST*, and the value indicates the overall ratio. The chart is in good agreement with the statistical analysis conducted above. The performance of *Greedy1* and *Greedy2* is similar, with a slight advantage going to *Greedy2*, as confirmed by the Wilcoxon test. *GA* achieves the best performance, and is significantly better than *SWO*, which in turn is significantly better than the greedy algorithms. The performance of the various algorithms appears to converge around the 10% density level, and approximates that of *KYST*.

Table 5.4: Friedman test results for the grid density grouping

density	Greedy1	Greedy2	SWO	GA
100%	9.70	9.72	9.74	9.74
90%	8.88	8.86	8.88	8.82
80%	8.14	8.16	8.14	8.05
70%	7.07	7.05	7.05	6.88
60%	6.04	6.04	6.08	5.89
50%	5.04	5.06	5.09	5.13
40%	4.08	4.10	4.01	4.03
30%	3.04	3.00	3.02	2.72
20%	2.02	2.02	2.00	2.11
10%	1.00	1.00	1.00	1.64
χ_r^2 [a]	466.10	467.21	470.19	433.88
W [b]	0.977	0.979	0.986	0.910

[a]χ_r^2 is Friedman's chi-square statistic; the critical value at $p = 0.001$ level is 27.88
[b]W is Kendall's coefficient of concordance; the critical value at $p = 0.001$ level is 0.058
The figures in rows 2–11 are mean ranks across 53 observations

Grid Density Analysis

Now we consider *Greedy1*, *Greedy2*, *SWO*, and *GA* separately. For each we have a block with 53 groups of 10 measurements corresponding to different levels of grid density. Table 5.4 reports the results of four Friedman tests with the null hypothesis H_0 of no difference between grid densities with regard to Stress (3.11), and the alternative hypothesis H_1 of at least one density having a different ranking from the others. The values of both Friedman's chi-square statistic and Kendall's coefficient of concordance are statistically significant at a level exceeding 0.001 for each algorithm, which allows H_0 be rejected in favour of H_1. Moreover, the coefficient of concordance is very close to its maximum value of 1, implying that there is an almost complete agreement between the 53 cases, especially for *Greedy1*, *Greedy2*, and *SWO*. Once H_0 has been rejected, post-hoc testing in the form of Tukey Multiple Comparisons can be applied, to identify significant differences between grid density rankings in pairs.

The outcome of the post-hoc test for *Greedy1*, *Greedy2*, and *SWO* is identical. All pairwise comparisons are significant at the 5% level, except for all adjacent density levels and $(100\%, 80\%)$, $(90\%, 70\%)$. However, the Wilcoxon test for each of these pairs shows the difference in ranking to be significant at a level exceeding 0.001. These results imply that there is a consistent gradation of performance for these algorithms depending on grid density, as can be witnessed from the observed ranks in columns 2-4 of Table 5.4, and the high value of Kendall's coefficient of concordance.

Tukey comparisons for all grid density pairs of the *GA* algorithm are significant at the 5% level, except for all adjacent density levels and (100%, 80%), (70%, 50%), (60%, 40%), and (30%, 10%). Again, the Wilcoxon test shows each of these differences to be significant at the $p < 0.001$ level. Thus, the performance of *GA* also depends on grid density; however, for densities of 30% and below the last column of Table 5.4 demonstrates the relationship to be less consistent between all the data collections. This effect is due to the fact that *GA* is a stochastic minimisation heuristic, and can produce suboptimal solutions. Therefore, there are a number of cases where a test measurement is inferior to one for higher density. This situation is more likely for sparse grids, which put small constraints on the configuration, and the corresponding global minima of Stress are close. A solution to this problem would be to run the algorithm a number of times, as we do for *KYST*, and use the best result. However, we did not want to give an unfair advantage to *GA* over the other algorithms, especially when taking into account the necessary increase of the computational effort required to complete the experiment.

Complete Analysis

To complete the analyses we present the results for a block of 53 groups of all 40 measurements. Table 5.5 details the outcome of the Friedman test, with the mean ranks of the various combinations of algorithm and density shown in descending order. Unsurprisingly, the values of both Friedman's chi-square statistic and Kendall's coefficient of concordance are statistically significant at a level exceeding 0.001, allowing the null hypothesis of no difference between conditions with regard to Stress (3.11) to be rejected. There are too many possible pairs of conditions for a Tukey Multiple Comparison procedure to be carried out. However, it would not yield any new insights in addition to the previous two analyses.

Table 5.5: Friedman test results for the complete grouping

density	algorithm	mean rank
100%	Greedy1	39.04
100%	Greedy2	38.77
90%	Greedy1	37.30
90%	Greedy2	36.64
80%	Greedy1	35.52
80%	Greedy2	34.76
100%	SWO	33.57
70%	Greedy1	32.07
70%	Greedy2	31.59
90%	SWO	30.90
80%	SWO	28.88
60%	Greedy1	28.62
100%	GA	28.06
60%	Greedy2	27.91
70%	SWO	25.58

Table 5.5: (continued)

density	algorithm	mean rank
90%	GA	24.29
50%	Greedy1	23.96
50%	Greedy2	23.28
60%	SWO	22.32
80%	GA	21.65
40%	Greedy1	19.05
50%	SWO	18.66
70%	GA	18.56
40%	Greedy2	18.35
60%	GA	15.57
40%	SWO	14.88
30%	Greedy1	13.84
50%	GA	13.45
30%	Greedy2	12.93
30%	SWO	10.51
40%	GA	10.39
20%	Greedy1	8.63
20%	Greedy2	7.84
30%	GA	7.11
20%	SWO	6.34
20%	GA	5.75
10%	GA	4.32
10%	Greedy1	3.70
10%	Greedy2	3.22
10%	SWO	2.20
	χ_r^2 [a]	1941.28
	W [b]	0.939

[a] χ_r^2 is Friedman's chi-square statistic; the critical value at $p = 0.001$ level is 72.05
[b] W is Kendall's coefficient of concordance; the critical value at $p = 0.001$ level is 0.035

Greedy1 always appears before *Greedy2* in Table 5.5, meaning that it has a higher overall rank, and therefore Stress, at each level of density. This is in agreement with the algorithm analysis. *SWO* at 100% density performs better than *Greedy2* at 80%, the same relation holds for 90 and 70%, 70 and 60%, 60 and 50% respectively. At 40% density and below *Greedy2* closes in on *SWO*, but always performs worse. Again, these observations are consistent with the algorithm analysis, and Figure 5.4 in particular. *GA* at 100% density is better than *SWO* at 80%; this relation also holds for 90 and 70%, 80 and 60%, 70 and 50%, 50 and 40%, 40 and 30% respectively. *GA* comes best at 20 and 30% density, but is the worst performer at 10%. The last observation does not agree with the algorithm analysis, which takes precedence as it is based on pairwise statistical tests with a documented confidence level.

There is a clear division of algorithmic complexity between the greedy algorithms, *SWO*, and *GA*. The time taken to execute the entire test is approximately 35 seconds, 2 hours 17minutes, and 80 days respectively[‡]. This translates to a ratio of 0.004 to 1 to 800. The first number follows from the fact that *SWO* is equivalent to repeating a simplified version of *Greedy2* 1000 times (*SWO.iterationlimit* parameter), and considering that initialisation costs dominate the simple computation of *Greedy1* and *Greedy2*. *GA* is almost three orders of magnitude slower than *SWO* because of the sheer number of Stress (3.11) evaluations needed for the population of solutions to converge.

5.3.2 Qualitative Evaluation

In Section 5.3.1 we have been able to systematically rank the algorithms based on their performance in terms of loss function (3.11). However, it is also important to show the extent to which proximity grids calculated by the algorithms differ visually, for if there is hardly any difference between them subjectively then the objective results are of little significance. We have selected a sample data collection from the test bed for this purpose. It is a graph with 16 vertices and 21 edges, which appears in Section B.1 of the appendices under the name *network*.

The dissimilarity between a pair of vertices in a graph is taken to be proportional to their graph theoretic distance – shortest path length (see Section 1.2.6) – so that adjacent vertices are the most similar of all pairs of vertices. Thus, a configuration with a low value of Stress (3.11) will have uniform edge lengths, and will preserve the graph topology well: Figure 5.12 is a drawing of the sample graph generated by *KYST*. However, the number of edge crossings, another important graph drawing aesthetic [Battista94], will not be minimised in general, as it might be in conflict with the previous criteria. There is some correlation between these aesthetics, since configurations with high Stress also exhibit a high degree of edge and vertex crossover, as illustrated by Figures 5.5 and 5.6.

As the basis for our qualitative evaluation we are using the experimental setup of Section 5.3.1. Since the graph is small, there are only 7 distinct grid lengths at the 10% density increments: 4, 5, 6, 7, 8, 9, and 13; Figures 5.5–5.11 present the respective configurations for all algorithms. Visually *Greedy1* and *Greedy2* are very messy, with a lot of crossover and a poor preservation of topology for high densities. For sparse grids these algorithms produce similar or identical results to *SWO*. Generally, *GA* and *SWO* configurations are comparable, but the advantage goes to the former for spreading graph drawings across the grid, and hence making better use of available space. The continuous MDS configuration in Figure 5.12 is rotated to its principal axes (see Section 4.3), and thus elongated in the horizon-

[‡]the time taken by *KYST* to generate continuous configurations, which constitute a starting point for the greedy algorithms and *SWO*, has not been included, as it is negligible when compared to the time required to run *GA*

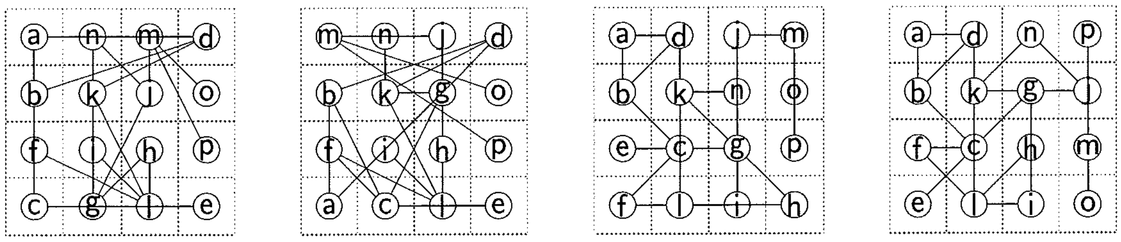

(a) *Greedy1* (b) *Greedy2* (c) *SWO* (d) *GA*

Figure 5.5: 4 × 4 proximity grids, 100% density. $\sigma(\mathit{Greedy1}) = 0.2173$, $\sigma(\mathit{Greedy2}) = 0.2257$, $\sigma(\mathit{SWO}) = 0.1246$, $\sigma(\mathit{GA}) = 0.095$

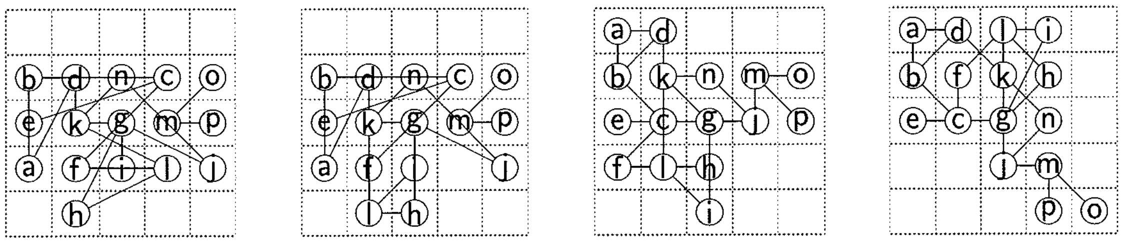


Figure 5.6: 5 × 5 proximity grids, 64% density. $\sigma(\mathit{Greedy1}) = 0.1632$, $\sigma(\mathit{Greedy2}) = 0.1345$, $\sigma(\mathit{SWO}) = 0.0566$, $\sigma(\mathit{GA}) = 0.0449$

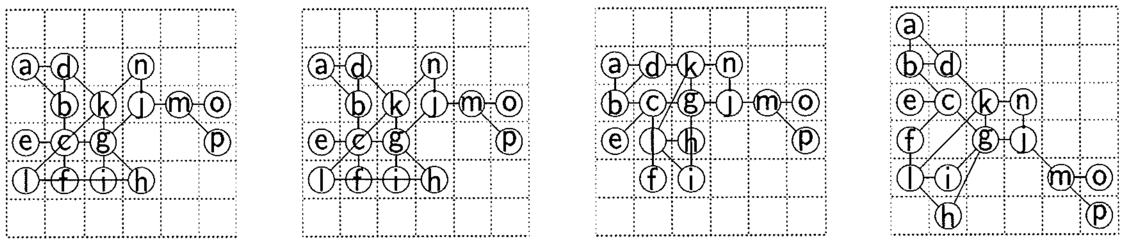


Figure 5.7: 6 × 6 proximity grids, 44% density. $\sigma(\mathit{Greedy1}) = \sigma(\mathit{Greedy2}) = 0.0478$, $\sigma(\mathit{SWO}) = 0.0371$, $\sigma(\mathit{GA}) = 0.0344$

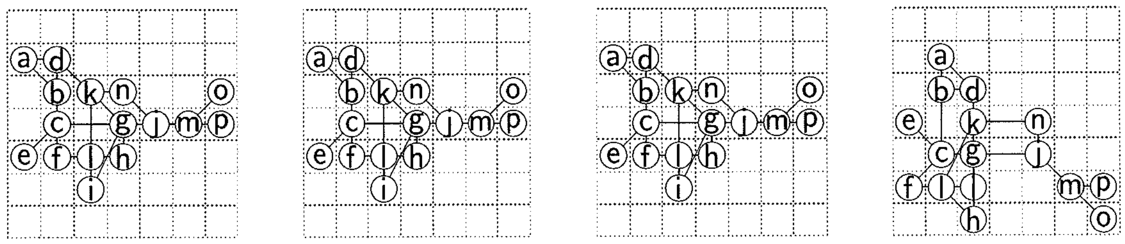


Figure 5.8: 7 × 7 proximity grids, 33% density. $\sigma(\mathit{Greedy1}) = \sigma(\mathit{Greedy2}) = \sigma(\mathit{SWO}) = 0.0323$, $\sigma(\mathit{GA}) = 0.0298$

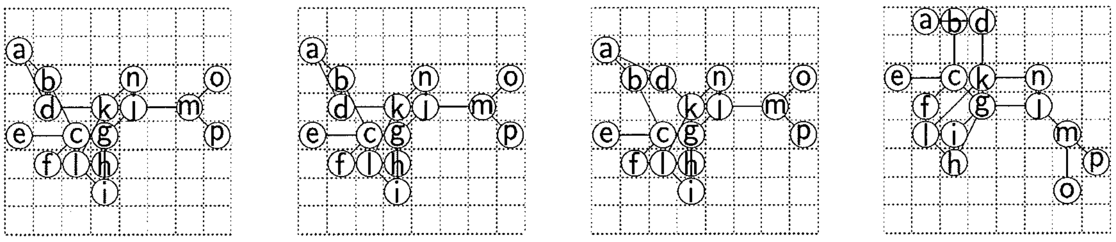


Figure 5.9: 8 × 8 proximity grids, 25% density. $\sigma(\textit{Greedy1}) = \sigma(\textit{Greedy2}) = 0.0339$, $\sigma(\textit{SWO}) = 0.0313$, $\sigma(\textit{GA}) = 0.0294$

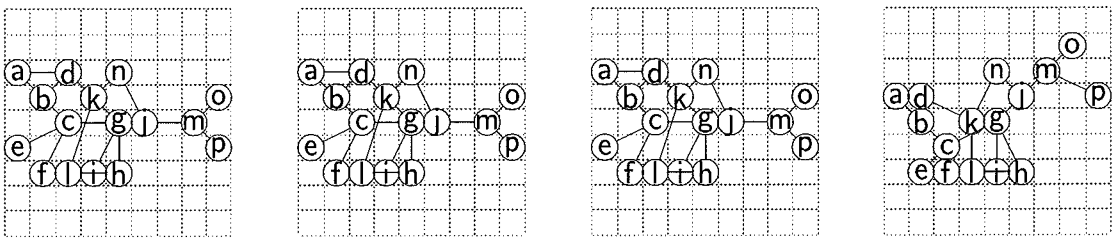


Figure 5.10: 9 × 9 proximity grids, 20% density. $\sigma(\textit{Greedy1}) = \sigma(\textit{Greedy2}) = \sigma(\textit{SWO}) = 0.0272$, $\sigma(\textit{GA}) = 0.0261$

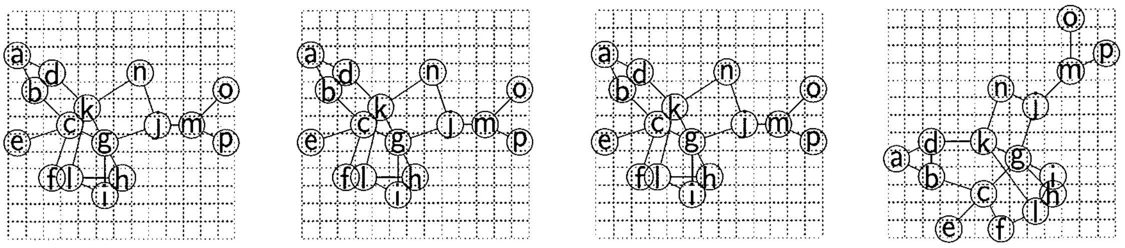


Figure 5.11: 13 × 13 proximity grids, 9% density. $\sigma(\textit{Greedy1}) = \sigma(\textit{Greedy2}) = \sigma(\textit{SWO}) = 0.0245$, $\sigma(\textit{GA}) = 0.0229$

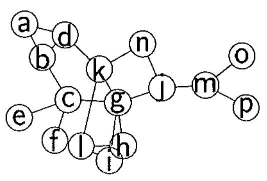


Figure 5.12: Continuous MDS configuration. $\sigma = 0.0209$

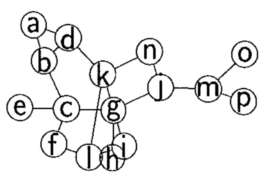


Figure 5.13: 13 × 13 proximity grid for the *GA* algorithm rotated with Procrustes Analysis to fit the MDS configuration

Table 5.6: Assessment of the sample graph

grid length	Greedy1	Greedy2	SWO	GA
4	23	30	4	4
5	16	13	2	4
6	4	4	3	1
7	2	2	2	2
8	3	3	2	2
9	2	2	2	2
13	2	2	2	2
rank total[a]	16	16	8	8

(a) number of edge and vertex crossovers

grid length	Greedy1	Greedy2	SWO	GA
4	0.2173	0.2257	0.1246	0.0950
5	0.1632	0.1345	0.0566	0.0449
6	0.0478	0.0478	0.0371	0.0344
7	0.0323	0.0323	0.0323	0.0298
8	0.0339	0.0339	0.0313	0.0294
9	0.0272	0.0272	0.0272	0.0261
13	0.0245	0.0245	0.0245	0.0229
rank total	19	19	14	7

(b) Stress

[a]the rank total for a given algorithm is the sum of its relative ranking over all grid lengths; if two or more algorithms are tied at a given grid length they all share the lowest available rank, and the next available rank is incremented by the number of tied algorithms less one, i.e. as if the tie had not occurred
The rank of a value is denoted in greyscale

tal direction. Since *SWO* and the greedy algorithms use this configuration as a starting point, they are unable to significantly alter its shape. The grid drawings at the 10% density level, including the *GA* one when factoring out the rotation (see Figure 5.13), very closely resemble the continuous configuration, with small differences in Stress between all of them.

Table 5.6(a) conveniently summarises the number of edge and vertex crossovers for all combinations of grid length and algorithm. The performance of both *SWO* and *GA* is very good, with very few crosses even for dense grids. They achieve an identical rank total, calculated by summing their respective ranking over all grid lengths. *Greedy1* and *Greedy2* also have the same rank total between them, however, they do not cope with dense grids well, and produce a large number of

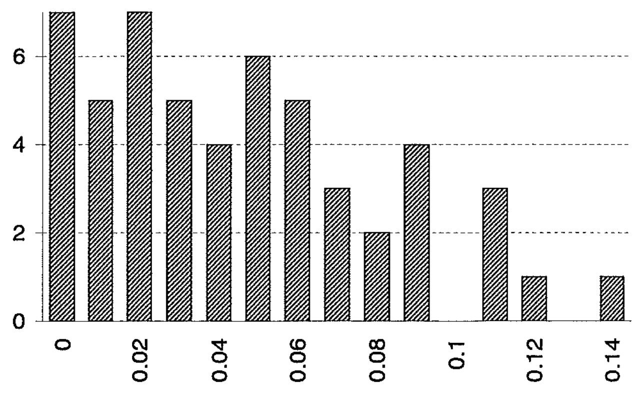


Figure 5.14: Histogram of Stress minima calculated by *KYST* for all collections in the test bed

crosses. Table 5.6(b) summarises Stress for all configurations, which is indicative of how well graph topology is preserved. *GA* is consistently the best, closely followed by *SWO*. The results for *Greedy1* and *Greedy2* are similar, and they achieve the same rank total. For dense grids there is an enormous difference between these two pairs of algorithms, which gradually reduces, and diminishes completely for sparse grids. These quantitative results are in agreement with the impressions reported in the previous paragraph.

5.4 Discussion

The data collections contained in the test bed are varied in type, size, and complexity. Stress (3.11) is normalised for the size of a collection (see Section 3.2.3), and its value at a minimum will depend on the inherent dimensionality or complexity of data. Figure 5.14 is a histogram of the test results for *KYST*, with each measurement rounded to the nearest bucket. A high value of Stress is indicative of high dimensionality of data, and suggests that a two-dimensional representation is not very faithful. Such data is seldom found in practice, and the small number of data collections with Stress over 0.1 reflects this. Other Stress values are fairly evenly distributed, which indicates the test bed to be a representative sample from a population of small-to-medium sized data collections. The size of the sample is large enough to warrant the use of statistical inference, and allows the results to be generalised to the whole population.

Both objective and subjective evaluations identify a significant difference between the greedy (*Greedy1*, *Greedy2*) and iterative (*SWO*, *GA*) algorithms, and confirm an intuitive expectation that the more computational effort is put in the better the results are. Statistical inference demonstrates *Greedy2* to be the better of the greedy algorithms, which again is to be expected, as an exact rather than an approximate method for finding the closest empty cell is used. *GA* generates the best proximity grids in terms of both objective and subjective quality, but is closely matched by *SWO* for sparser grids at a fraction of the computational cost.

Dense proximity grids provide visualisations with the highest information density, since there are none or very few empty areas. At the same time stringent constraints are imposed on the preservation of object proximity, and the use of a global optimisation heuristic, for example GA, is essential to find a good allocation of objects to cells. Grids of medium density (20-70%) achieve a useful amount of separation, and allow fairly large object representations with no overlap: see Figure 5.1(b) for an example of a 70% proximity grid. The increase in Stress (3.11) over a continuous MDS configuration is moderate, and a hybrid technique like SWO is capable of matching global optimisation for quality. A sparse grid is not a very large departure from the corresponding continuous configuration, and a greedy approach is capable of transforming the latter to the former. Alternatively, a hybrid technique can be run for a small number of iterations, to overcome any potential misallocation. Sparse grids only give a limited amount of separation, but nonetheless can be useful for configurations with complete overlap or clumps of points.

Obviously, separation between icons can be increased by enlarging the canvas area, while keeping the icon size fixed. However, GA will use available space on a given canvas most efficiently, i.e. require partition into the smallest number of cells. SWO will need a larger grid size to provide an equally accurate representation of proximity relationships; the sparsest grid will be attributed to the greedy algorithm, in general. Thus, when populating the canvas with icons, GA will afford the largest icon size without overlap. In a particular application, the information density provided by each method has to be weighed against its computational requirement, the two criteria being in complete conflict.

5.5 Related Work

A Self-Organising Map (SOM) [Kohonen89] is a neural network that learns a mapping from an n-dimensional space to an m-dimensional lattice, which preserves proximity relations. The network is comprised of two fully connected layers: n continuous-valued inputs, and a lattice, usually two-dimensional, of output units. The training consists of presenting successive input tuples, and updating the winner neuron, i.e. the output unit with the weight vector closest to the current input, as well as its neighbours. Once the training has been completed, the mapping can be read out by enumerating every input, and noting the lattice coordinates of the winner neuron. With a sufficient number of output units for a given input set, this algorithm can generate a one-to-one mapping between them, i.e. a proximity grid. However, this method will not work if identical or nearly identical tuples are to be represented separately. A SOM can only process a data table with quantitative attributes (see Section 1.2.1), and so is not as widely applicable as the algorithms presented in this chapter.

Chapter 6

Case Studies

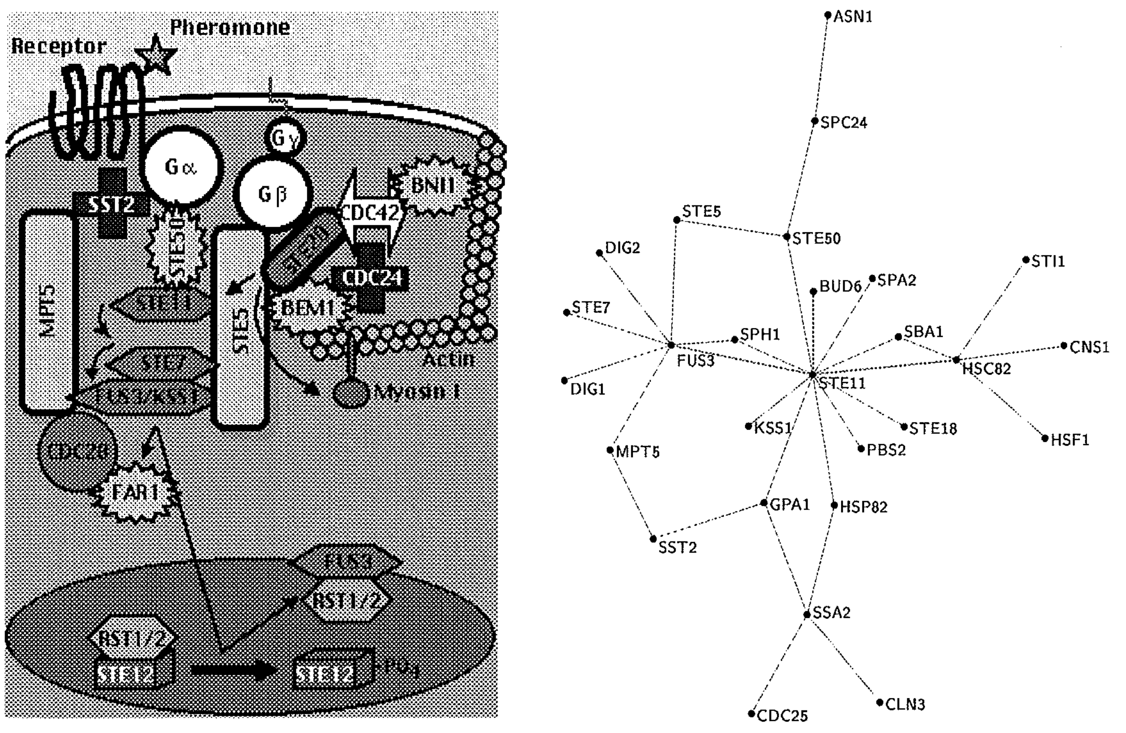
Three applications of proximity visualisation are presented in this chapter, which tackle disparate areas, and give a testament to how generic this visualisation paradigm is. Two of the case studies are the result of collaboration on analysing protein interactions and evaluating an interface for image browsing. The final one is more of a proposal of how a database and its metadata could be explored effectively by means of proximity visualisation.

6.1 Protein Interactions

The topic of this section is an application of proximity visualisation to collections of protein interactions. Only a brief summary and a single example are presented here. A complete treatment of the problem of visualising protein interactions has been published separately [Basalaj99], as a result of a collaboration with Karen Eilbeck – a bioinformatics researcher at the University of Manchester.

The analysis of protein-protein interactions within a cell is concerned with understanding the function of individual proteins and networks of proteins. The presence of an interaction between a pair of proteins implies the existence of a binary relationship. There are various types of interactions, but the ones we analysed are all bidirectional. As discussed in Section 1.2.6, such a collection of entities and relationships between them can be expressed as a graph, which is undirected in this case. Thus, the shortest path length between a pair of proteins in the graph is taken as their dissimilarity.

A study of a protein interaction cluster in yeast, a unicellular organism, is presented here. This cluster is of great interest to pharmaceutical companies, as it represents a signal transduction pathway that is responsible for passing stimuli from the cell membrane to its nucleus. This process is initiated by a membrane-bound receptor, a G-protein, GPA1 [Rens-Dom95]. Figure 6.1(a) is a diagram put together by a domain expert of this network of interacting proteins, with arrows to represent both enzymatic mechanisms and movement. Figure 6.1(b) is a proximity visualisation of the same protein cluster, with interactions gathered from biochemical and library based experiments. In general, only the proteins



(a) expert (b) proximity visualisation

Figure 6.1: Comparison of protein interaction drawings

relevant to the transduction mechanism are common to both drawings, and other proteins might not have a counterpart in the other drawing.

Overall, Figure 6.1(b) captures more information, though a few interactions are missing as a result of experimental errors. The essential edges are: GPA1 interacts with SST2, which in turn interacts with MPT5; and the cascade STE11, STE7, FUS3/KSS1. Unfortunately, the interaction between STE11 and STE7 is missing, but the rest of the architecture is there: STE5, STE50, and STE11. According to biologists that inspected both drawings the correlations between them are impressive.

Figure 6.1(b) has been created with the *hybrid* algorithm of Section 3.5.5 by minimising Energy (3.13). It follows from the argument of Section 3.2.4 that any deviation in distance between adjacent vertices is penalised more than the same error for non-adjacent vertices. Therefore, a straight-line drawing based on an optimal vertex layout will have uniform edge lengths. At the same time an even spread of vertices is achieved, because the distance between a non-adjacent pair of vertices tends to be proportional to their dissimilarity, i.e. their shortest path length. Overall, the dissimilarities between vertices will be preserved as closely as possible, and hence the topology of the graph. Multidimensional scaling has been used for generating graph drawings before [Kruskal78a, Kamada89, Cohen97]. It is worth noting that this method does not explicitly minimise edge crossings or maximise drawing symmetry, other important graph drawing aesthetics [Battista94], but still achieves pleasing results.

There are many alternative algorithms for drawing graphs [Battista94]; however, the advantage of using MDS for visualising protein interactions is that it is not specific to a graph representation of these data. Instead of a dissimilarity coefficient based on graph theoretic distances, one that takes the location of proteins within the cell into account can be used, for example. A comparison of configurations resulting from these two representations might help to clarify the relationship between the location of a protein and its interactions.

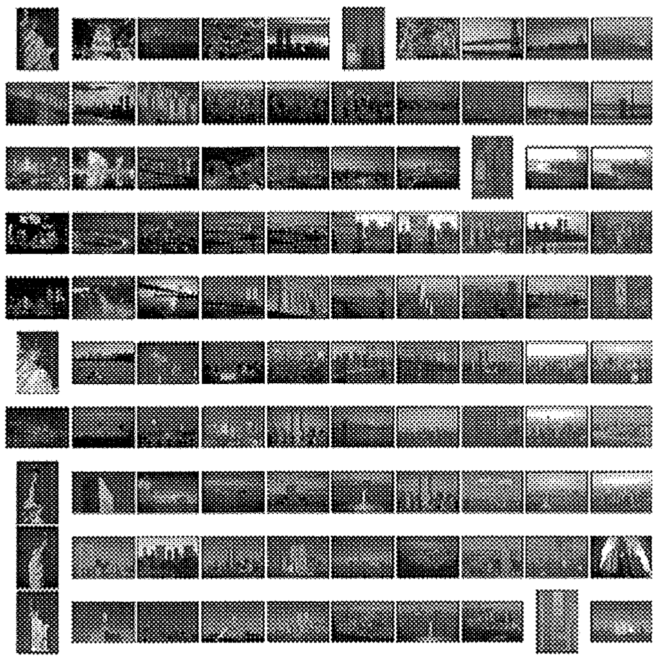
It is customary for collections of protein interactions to be drawn manually, or be presented in a tabular form. However, these approaches are not practical for anything but small collections, and automatically generated two- or three-dimensional graph drawings are a welcome alternative. To our knowledge, it is the first time a graph drawing technique has been applied in this context. With typically large quantities of protein interactions in most collections, visual representation is an ideal method to communicate such complex information within the biological community. Moreover, proximity visualisations are invaluable for discovering data inconsistencies, and carrying out comparative studies between organisms [Basalaj99].

6.2 Image Browsing

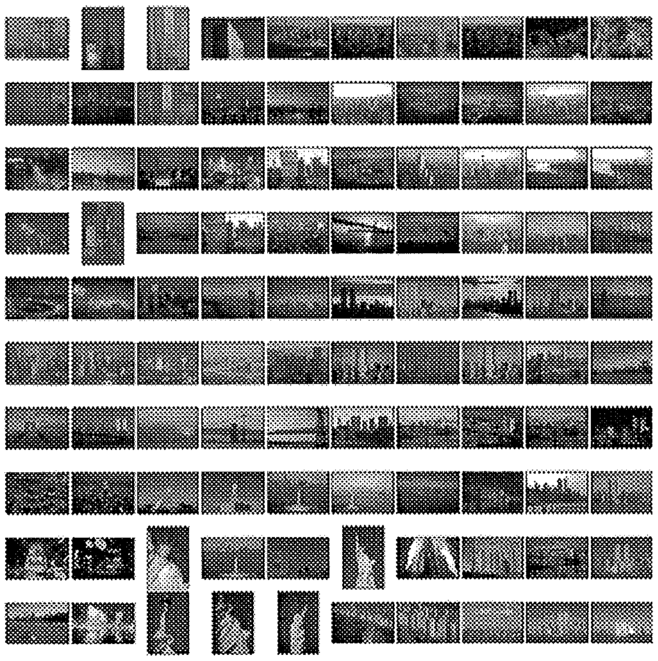
Proximity visualisation can be applied to multimedia collections, for example collections of images or text documents. This case study is based on the results of joint work with Kerry Rodden of University of Cambridge Computer Laboratory, and also David Sinclair and Kenneth Wood of AT&T Laboratories in Cambridge. A series of experiments have been undertaken to verify the usefulness of proximity visualisation as a tool for image browsing [Rodden99a, Rodden99b].

Thanks to improvements in storage and networking technologies, large digital collections of images are now commonplace. To facilitate access to these collections, it is necessary to perform some indexing of their content. If textual annotations exist it is possible to use conventional information retrieval methods, for example the vector model of Section 1.2.8. However, annotating images is a time-consuming process, and the results tend to be highly subjective. An alternative is to classify images based directly on their content. The most common approach is to use low-level visual features of images such as colour and texture, extracted by the IRIS similarity coefficient of Section 1.2.7, for instance.

Forming a visual query to an image collection is considerably more difficult than a textual query. The user is required to either sketch a prototype image or select a suitable example by browsing the collection. The query result will be a set of images judged by the system to be the most similar. This is based, however, on the low-level visual features, and thus might disagree with the user's perception. There can be many images satisfying a query, especially if it is not well formed. These difficulties make provision of good support for browsing paramount.



(a) visual similarity



(b) caption similarity

Figure 6.2: 100 images of New York arranged in a 10 × 10 proximity grid by visual or caption similarity. These images have been taken from the Corel stock photograph collection

A proximity visualisation of an image collection, with each individual image represented by a thumbnail, appears to constitute a good paradigm for a browsing interface. To prevent thumbnails from overlapping one another, they can be arranged in a proximity grid. Figure 5.1(b) on page 72 serves as an example, arranging 100 images of Kenya by visual similarity. The combination of the image similarity coefficient and proximity grid has uncovered a natural structure in the collection. Photographs of people are clustered in the top right corner, buildings are to the left, and wildlife is at the bottom. Existing image browsing tools commonly employ grid arrangements, as they assist in systematic scanning of a collection; however, thumbnails are positioned arbitrarily or chronologically, instead of according to their similarity.

Figure 6.2(a) is another example of an image collection, 100 photographs of New York this time, arranged by visual similarity in a proximity grid. This display makes it very easy to locate photographs with similar colour composition, e.g. sunsets in the top right corner, night time photos at the top left, and panoramas at the bottom. Each image has a caption associated with it, describing its main features in a few keywords. Thus, it is thus possible to give an alternative view of the collection, with photographs arranged by caption similarity. All thematically related images are grouped in Figure 6.2(b), e.g. photographs of the Statue of Liberty. Both views are complementary, as they support different types of browsing tasks.

6.3 Databases

Visualisation of relational and object databases poses a number of challenges related to the complexity that needs to be anticipated in data. Such databases are usually heterogeneous, requiring either a visualisation method that is generic enough to handle all possible data types, or a set of specific methods that cater for the individual combinations of data types occurring in the database. Both these approaches can be combined, with the generic method used whenever a specific one is unavailable or unnecessary.

Because of this inherent structural complexity, a database schema (metadata) deserves to be visualised in its own right. A preliminary metadata analysis can assist in planning and structuring a subsequent data analysis. There is an enormous benefit in integrating both forms of visualisation, because the metadata presentation provides context for exploring and navigating through detail presentations of data. Although a database is likely to change over time, its schema, if properly defined to start with, will undergo few changes. Having a stable starting point for the visualisation process is important, and metadata presentation is an ideal choice.

Integration of metadata visualisation, to convey the overall structure of a given database, and data visualisation, to reveal the actual detail, enables the database to be analysed in its entirety. Most existing database visualisation systems, e.g. DataSpace [Petajan97], Exbase [Lee96], VizDB [Keim94], focus solely on data visualisation, disregarding the benefits of metadata visualisation. Such an approach is satisfactory for a homogeneous database, but fails in the more general and common heterogeneous case.

6.3.1 Metadata Visualisation

A database schema can be graphically represented as an Entity-Relationship diagram. An entity, represented as a box in the diagram, is a type that is defined in the schema. A relationship, shown as a link between boxes, pertains to a pair of entities. Inheritance can be represented in a number of ways, for example by inclusion of a subtype entity within its supertype. Entity attributes can be represented either as elements of an entity box, or as separate boxes attached to it. An E-R diagram has a number of desirable properties: familiarity, naturalness, and scalability.

An E-R diagram is a graph, in essence, with vertices having complex properties and possibly being nested. Visualising it in a proximity grid will allow large visual representations of entities to be accommodated, without causing them to overlap. It is convenient to arrange that by interacting with this schema representation a viewer can drill down to the entity (type) instance level. At the same time the E-R diagram provides context for visualisation of this higher level of detail, and thus assists in exploration of the database. For example, a fisheye lens metaphor [Furnas86] can be employed, so that details of data and relationships can be explored, while maintaining the context at the periphery.

6.3.2 Data Visualisation

Proximity visualisation is well suited to representing collections of abstract objects. Dissimilarity between pairs of tuples from a relational database table, or objects from an object database extent can be assessed with coefficient (1.8), introduced on page 4, which allows a heterogeneous mix of attributes, with a possibility of conditionally defined or missing values. The method loses its appeal when strong attribute semantics are present, e.g. longitude and latitude in a Geographic Information System database, in which case a specialised visualisation method is likely to be superior. However, proximity visualisation is very effective in a general case, and as such would be a good default visualisation choice.

Figure 6.3 gives an example of the proximity visualisation of a database table, based on the *crcars* data table [Donoho83], which also appears in Section B.1 in the appendices. The following attributes of 406 cars have been recorded: model name, miles per gallon, number of cylinders, engine displacement, horsepower, weight, acceleration, model year, and origin – American, European, or Japanese; with some values missing. The Euclidean λ-general dissimilarity coefficient (1.8) has been used to calculate dissimilarities between cars in pairs. All attributes have been given equal weight, except model name and origin, which have been given zero weight, and thus effectively excluded from the calculation of dissimilarity for the same reasons as in Section 2.1.

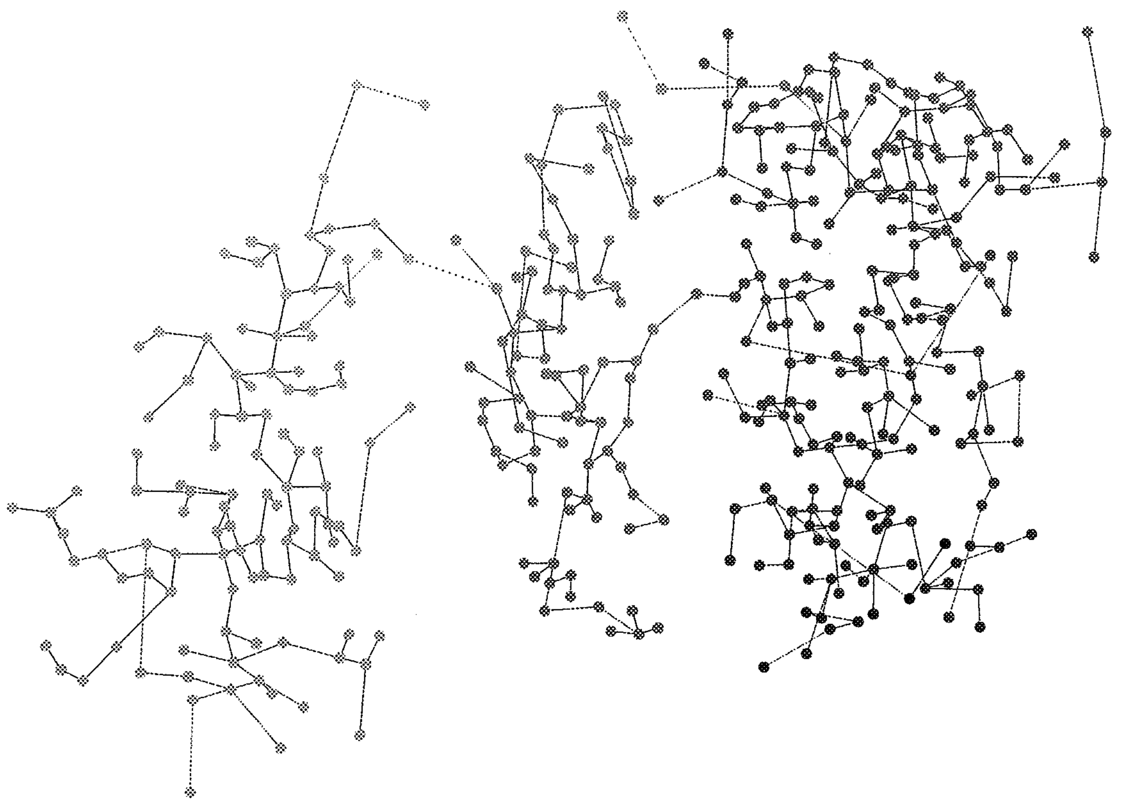


Figure 6.3: Proximity visualisation of the *crcars* data table

Individual cars are represented as solid circles in Figure 6.3, with the green colour component assigned to the number of cylinders attribute, and the red component linked to model year. These two attributes can be seen to explain the data well, with the former determining the horizontal location of cars, and the vertical direction coinciding with the latter attribute. Three clusters can be clearly identified, from left to right: 8, 5 or 6, and 4 cylinder cars. This structure is easy to explain once it is realised that the number of cylinders strongly determines other characteristics of a car, like its weight and horsepower, as do technological advances captured in the model year attribute.

A Minimum Spanning Tree (MST) can be computed for the *cars* data table, as in Section 4.2.1, to identify the most similar pairs of cars – leaves in a single-link cluster hierarchy. The MST is superimposed on the configuration of Figure 6.3, where the weight of an edge connecting a pair of cars is equal to their dissimilarity, and is coded in greyscale, ranging from black for an identical pair to light grey for the most dissimilar pair. In a perfect geometrical representation of the data, no edges would cross over, and every vertex (car) will be connected by an edge to its closest neighbour. Thus, overlying the MST on the configuration allows one to see easily which links are badly represented, and which vertices are positioned inaccurately [Gordon81]. Based on this criterion the visualisation of Figure 6.3 can be seen to have a small degree of error. Such a graph representation ties in naturally with the schema graph.

Proximity visualisation can be used to represent more than one database table or object extent a time. Potentially, if two sets of tuples or objects have some overlapping attributes it will be possible to calculate pairwise dissimilarities within the amalgamated set, and subsequently apply proximity visualisation. This could be an effective way of visualising relationships and inheritance at the detailed, instance level, analogously to drawing a bipartite graph.

On the other hand, proximity visualisation is also suitable for representing a subset of instances from a database table or an object extent, resulting from a query, for example. Thus, a viewer could select a subset of elements in a visualisation, in order to analyse the corresponding instances in more detail. Apart from a specialised view of data, this query-by-example mechanism can be used to construct a generalised view. Such a natural evolution of analysis, coupled with provision for backtracking, will encourage the viewer to explore data.

Chapter 7

Conclusions

Throughout the course of this dissertation Proximity Visualisation has been defined, and a number of related techniques that fulfil this paradigm have been demonstrated. A well established statistical technique of Multidimensional Scaling is the most fundamental of these, and has been applied to information visualisation before. The contribution of this work is a comprehensive survey of multidimensional scaling algorithms, an empirical assessment of their effectiveness, and a recommendation for the best, hybrid approach.

Multidimensional scaling is effective at visualising abstract data collections; however, it is computationally intensive, and becomes very time consuming for large collections of objects. Cluster analysis can identify the most important objects, so that more time can be devoted to representing them, over less structurally significant objects. An incremental multidimensional scaling procedure is proposed that has lower time and space requirements, achieved by gradually introducing objects into a visualisation based on their importance. As a result, it becomes practical to visualise data collections two orders of magnitude larger than with standard multidimensional scaling. The validity of this concept has been confirmed by means of a rigorous evaluation.

Proximity Grid is a novel visualisation technique, especially suited to the design of user interfaces, as it can provide a display with high information density. Icons representing objects from a data collection are arranged in a grid, and thus can occupy their respective cells completely, without overlap. Rather than assigning icons and the corresponding objects arbitrarily to grid cells, they are positioned so that proximity relationships between objects are preserved as well as possible. Unlike multidimensional scaling, this is a combinatorial problem, and different heuristics are needed for solving it. A number of algorithms of varied complexity have been presented, and the trade-off between their effectiveness and responsiveness established.

The evaluation framework, used in characterising algorithms for generating each type of proximity visualisation, is innovative in its own right. A number of real world data collections have been selected to serve as input for each algorithm under test. The objective quality of the resulting visualisations was recorded, so that a ranking of the algorithms could be established with statistical analysis. It would have been possible to present these series of measurements in the form of charts, and draw conclusions based on visual inspection, as is customary in such empirical studies. However, no matter how useful visualisation is, it cannot be applied blindly, and in this case the use of statistical inference has yielded a more concise and objective decision.

This dissertation is concerned with visualisation of abstract data, and is rich in examples as a consequence. A sample output of every algorithm has been provided, so that the extent and character of the differences between algorithms can be easily ascertained. Such a qualitative evaluation naturally complements the statistical analysis. Also, a number of case studies have been presented that illustrate the broad range of applications of proximity visualisation.

The overall thesis is that proximity visualisation is more generic than other information visualisation techniques, as any conceivable data type can be represented. Even heterogeneous data can be accommodated by means of the general dissimilarity coefficient. For data with strong temporal or spatial semantics a specialised technique is more natural, and likely to provide a superior visual representation. However, an abstract collection of objects is best described and visualised in terms of object proximity.

Appendix A

Partial Derivatives of Energy

For convenience we repeat the definition of raw Energy (3.16) here:

$$\sigma_{Er}(\boldsymbol{X}) = \sum_{i<j} \frac{\left(d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}\right)^2}{\hat{d}_{ij}^2} \tag{A.1}$$

and that of the Euclidean distance (3.1) between points $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$ and $\boldsymbol{x}_j$ in the p-dimensional MDS configuration $\boldsymbol{X} = [x_{ia}]$:

$$d_{ij}(\boldsymbol{X}) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{\sum_{a=1}^{p} (x_{ia} - x_{ja})^2} \tag{A.2}$$

A.1 The First Partial Derivative of the Euclidean Distance

$$\begin{aligned}
\frac{\partial d_{ij}(\boldsymbol{X})}{\partial x_{ka}} &= \frac{1}{2d_{ij}(\boldsymbol{X})} \frac{\partial d_{ij}^2(\boldsymbol{X})}{\partial x_{ka}} \\
&= \frac{1}{2d_{ij}(\boldsymbol{X})} \sum_{b=1}^{p} 2(x_{ib} - x_{jb}) \frac{\partial (x_{ib} - x_{jb})}{\partial x_{ka}} \\
&= \frac{x_{ia} - x_{ja}}{d_{ij}(\boldsymbol{X})} \frac{\partial (x_{ia} - x_{ja})}{\partial x_{ka}} \\
&= \frac{x_{ia} - x_{ja}}{d_{ij}(\boldsymbol{X})} (\theta_{ik} - \theta_{jk})
\end{aligned} \tag{A.3}$$

where $\theta_{rs} = \left\{ \begin{array}{lcl} 1 & : & r = s \\ 0 & : & r \neq s \end{array} \right.$ is the Kronecker delta

A.2 The First Partial Derivative of Energy

$$\frac{\partial \sigma_{Er}(\boldsymbol{X})}{\partial x_{ka}} = 2 \sum_{i<j} \frac{d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}}{\hat{d}_{ij}^2} \frac{\partial d_{ij}(\boldsymbol{X})}{\partial x_{ka}}$$

$$
\begin{aligned}
&= 2\sum_{i<j} \frac{d_{ij}(\boldsymbol{X}) - \hat{d}_{ij}}{d_{ij}(\boldsymbol{X})\hat{d}_{ij}^2}(x_{ia} - x_{ja})(\theta_{ik} - \theta_{jk}) \\
&= \sum \begin{cases} k = i < j \neq k & \Rightarrow \quad 2\displaystyle\sum_{j=l>k} \frac{d_{kl}(\boldsymbol{X}) - \hat{d}_{kl}}{d_{kl}(\boldsymbol{X})\hat{d}_{kl}^2}(x_{ka} - x_{la})(\theta_{kk} - \theta_{lk}) \\ k \neq i < j = k & \Rightarrow \quad 2\displaystyle\sum_{i=l<k} \frac{d_{kl}(\boldsymbol{X}) - \hat{d}_{kl}}{d_{kl}(\boldsymbol{X})\hat{d}_{kl}^2}(x_{la} - x_{ka})(\theta_{lk} - \theta_{kk}) \\ k \neq i < j \neq k & \Rightarrow \quad 0 \\ k = i < j = k & \Rightarrow \quad \text{contradiction} \end{cases} \\
&= 2\sum_{l>k} \frac{d_{kl}(\boldsymbol{X}) - \hat{d}_{kl}}{d_{kl}(\boldsymbol{X})\hat{d}_{kl}^2}(x_{ka} - x_{la}) - 2\sum_{l<k} \frac{d_{kl}(\boldsymbol{X}) - \hat{d}_{kl}}{d_{kl}(\boldsymbol{X})\hat{d}_{kl}^2}(x_{la} - x_{ka}) \\
&= 2\sum_{l\neq k} \frac{d_{kl}(\boldsymbol{X}) - \hat{d}_{kl}}{d_{kl}(\boldsymbol{X})\hat{d}_{kl}^2}(x_{ka} - x_{la}) \qquad \text{(A.4)} \\
&= 2\sum_{l\neq k} \frac{x_{ka} - x_{la}}{\hat{d}_{kl}^2} - 2\sum_{l\neq k} \frac{x_{ka} - x_{la}}{d_{kl}(\boldsymbol{X})\hat{d}_{kl}} .
\end{aligned}
$$

A.3 Second Partial Derivatives of Energy

$$
\begin{aligned}
\frac{\partial^2 \sigma_{Er}(\boldsymbol{X})}{\partial x_{ka}^2} &= 2\sum_{l\neq k} \frac{\theta_{kk} - \theta_{lk}}{\hat{d}_{kl}^2} - 2\sum_{l\neq k} \frac{1}{\hat{d}_{kl}} \frac{\partial \dfrac{x_{ka} - x_{la}}{d_{kl}(\boldsymbol{X})}}{\partial x_{ka}} \\
&= 2\sum_{l\neq k} \frac{1}{\hat{d}_{kl}^2} - 2\sum_{l\neq k} \frac{1}{\hat{d}_{kl}} \frac{d_{kl}(\boldsymbol{X})(\theta_{kk} - \theta_{lk}) - (x_{ka} - x_{la})\dfrac{\partial d_{kl}(\boldsymbol{X})}{\partial x_{ka}}}{d_{kl}^2(\boldsymbol{X})} \\
&= 2\sum_{l\neq k} \frac{1}{\hat{d}_{kl}^2} - 2\sum_{l\neq k} \frac{d_{kl}(\boldsymbol{X}) - \dfrac{(x_{ka} - x_{la})^2}{d_{kl}(\boldsymbol{X})}(\theta_{kk} - \theta_{lk})}{d_{kl}^2(\boldsymbol{X})\hat{d}_{kl}} \\
&= 2\sum_{l\neq k} \frac{1}{\hat{d}_{kl}^2} - 2\sum_{l\neq k} \frac{d_{kl}^2(\boldsymbol{X}) - (x_{ka} - x_{la})^2}{d_{kl}^3(\boldsymbol{X})\hat{d}_{kl}} \qquad \text{(A.5)}
\end{aligned}
$$

$$
\begin{aligned}
\underset{a\neq b}{\forall} \frac{\partial^2 \sigma_{Er}(\boldsymbol{X})}{\partial x_{ka} x_{kb}} &= 0 - 2\sum_{l\neq k} \frac{1}{\hat{d}_{kl}} \frac{\partial \dfrac{x_{ka} - x_{la}}{d_{kl}(\boldsymbol{X})}}{\partial x_{kb}} \\
&= -2\sum_{l\neq k} \frac{0 - (x_{ka} - x_{la})\dfrac{(x_{kb} - x_{lb})}{d_{kl}(\boldsymbol{X})}(\theta_{kk} - \theta_{lk})}{d_{kl}^2(\boldsymbol{X})\hat{d}_{kl}} \\
&= 2\sum_{l\neq k} \frac{(x_{ka} - x_{la})(x_{kb} - x_{lb})}{d_{kl}^3(\boldsymbol{X})\hat{d}_{kl}} \qquad \text{(A.6)}
\end{aligned}
$$

Appendix B

Test Bed

B.1 Small Data Sets

A small data set can either be a graph (g), image collection (i), dissimilarity matrix (m), or a data table (t).

Table B.1: Small data collections

name	size	type	source
aud	18	m	Mietta E. Lennes, Department of Phonetics, University of Helsinki
bridges	108	t	UCI Machine Learning Repository (MLR) [Blake98]
crcars	406	t	D. Donoho [Donoho83]
cars	38	t	H. V. Henderson [Henderso81]
gd98c	62	g	Graph Drawing '98 Contest
corel-244	100	i	Kerry Rodden, Computer Laboratory, University of Cambridge
corel-385	100	i	Kerry Rodden
cpu-performance	209	t	MLR
dermatology	366	t	MLR
detroit	13	t	StatLib Datasets Archive, http://lib.stat.cmu.edu/datasets/
echocardiogram	131	t	MLR
ecoli	107	g	Karen Eilbeck, Biochemistry Division, University of Manchester
FFT	18	m	Mietta E. Lennes
flags	194	t	MLR
gd99c	105	g	Graph Drawing '99 Contest
gene2sc	112	t	MLR
glass	214	t	MLR
GPA1component	27	g	Karen Eilbeck

Table B.1: (continued)

name	size	type	source
group	16	t	Opera Group, Computer Laboratory, University of Cambridge
haberman	306	t	MLR
heart	270	t	MLR
house-votes	435	t	MLR
housing	506	t	MLR
humandevel	130	t	MLR
image	210	t	MLR
imports-85	205	t	MLR
ionosphere	351	t	MLR
iris	150	t	MLR
Kellog	23	t	T. Cox [Cox94]
letters	26	t	F. Labelle's Dimensionality Reduction page, http://www.cs.mcgill.ca /~sqrt/dimr/dimreduction.html
letters-back	24	t	F. Labelle
liver-disorders	345	t	MLR
misc	206	t	MLR
network	16	g	J. B. Kruskal [Kruskal78a], Figure 3: "Input for Corn Biomass Network"
odmg-schema	22	g	The Object Database Standard version 2.0 [Cattell97]
o-ring-erosion	23	t	MLR
places	329	t	StatLib
planets	9	t	F. Labelle
post-operative	90	t	MLR
protein	25	t	Handbook of Small Data Sets [Hand94]
query	20	t	MLR
retention	270	t	U.S. Department of Agriculture, http://www.nal.usda.gov /fnic/foodcomp/Data/
servo	167	t	MLR
shuttle	15	t	MLR
Skulls	40	t	T. Cox
solar-flare	323	t	MLR
soybean	307	t	MLR
tae	151	t	MLR
usa-sales	26	t	American Automobile Manufacturers' Association (AAMA), http://www.economagic.com/aama.htm
usa-share	26	t	AAMA

Table B.1: (continued)

name	size	type	source
wine	178	t	MLR
Yoghurt	12	t	T. Cox
zoo	101	t	MLR
minimum	9		
maximum	506		
mean	145		
median	107		

B.2 Medium Data Sets

All medium-sized data sets in the test bed are tabular, and come from the UCI Machine Learning Repository[Blake98].

Table B.2: Medium-sized data tables

name	size
australian	690
breast-cancer	699
cloud-1	1024
cloud-2	1024
cmc	1473
credit-screening	690
german	1000
pima-indians-diabetes	768
tic-tac-toe	958
vehicle	846
vowel	990
water-treatment	527
yeast	1484
minimum	527
maximum	1484
mean	936
median	958

B.3 Large Data Sets

All large data sets in the test bed are tabular. The number of attributes (columns) is also given, as it is relevant for Principal Components Analysis (see Section 4.3), which was applied to them.

Table B.3: Large data tables

name	rows	cols	source
abalone	4177	8	UCI Machine Learning Repository (MLR) [Blake98]
letter	20000	16	MLR
mfeat	2000	6	MLR
pageblock	5473	10	MLR
sat	6435	36	MLR
segment	2310	18	MLR
nbody-1	15000	12	Jarrod Hurley, Institute of Astronomy, University of Cambridge
nbody-2	14898	12	Jarrod Hurley
shuttle	58000	9	MLR
spambase	4601	57	MLR
minimum	2000		
maximum	58000		
mean	13289		
median	5954		

Bibliography

[Anderber73] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[Andrews72] D.F. Andrews. Plots of high dimensional data. *Biometrics*, 28:125–136, 1972.

[Baeza-Ya99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, Harlow, 1999.

[Basalaj99] W. Basalaj and K. Eilbeck. Straight-line drawings of protein interactions. In *Proc. of Graph Drawing '99*, volume LNCS 1731, pages 259–266, Stirin, Czech Republic, September 1999.

[Battista94] G. Di Battista et al. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, 4:235–282, 1994.

[Blake98] C. L. Blake and C. J. Merz. *UCI Repository of machine learning databases.* University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

[Borg97] I. Borg and P. Groenen. *Modern Multidimensional Scaling.* Springer-Verlag, New York, 1997.

[Brandenb95] F. J. Brandenburg, M. Himsolt, and C. Rohrer. An experimental comparison of force-directed and randomized graph drawing algorithms. In *Proc. of Graph Drawing '95*, volume LNCS 1027, pages 76–87, Passau, Germany, September 1995.

[Busing97] F. M. T. A. Busing, J. J. F. Commandeur, and W. J. Heiser. *PROXSCAL: A Multidimensional Scaling Program for Individual Differences Scaling with Contraints.* SOFTSTAT, 1997.

[Cattell97] R. G. G. Cattell et al. *The Object Database Standard: ODMG 2.0.* Morgan Kaufmann Publishers, Los Altos, California, 1997.

[Chalmers96] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *Proc. of Visualization '96*, pages 127–132, San Francisco, October 1996.

[Chernoff73] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

[Clevelan84] W. S. Cleveland and R. McGill. The many faces of the scatterplot. *Journal of the American Statistical Association*, 79:807–822, 1984.

[Cohen97] J. D. Cohen. Drawing graphs to convey proximity: An incremental arrangement method. *ACM Transactions on Computer-Human Interaction*, 4:197–229, September 1997.

[Cox94] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling.* Chapman & Hall, London, 1994.

[deJong75] K. A. de Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[deLeeuw77] J. de Leeuw. Applications of convex analysis to multidimensional scaling. In J. R. Barra et al., editors, *Recent developments in statistics*, pages 133–145. North-Holland, Amsterdam, 1977.

[deLeeuw84] J. de Leeuw and I. Stoop. Upper bounds of Kruskal's Stress. *Psychometrika*, 49:391–402, 1984.

[Donoho83] D. Donoho and E. Ramos. PRIMDATA: Data sets for use with PRIM-H. In *American Statistical Association (ASA) Second Exposition of Statistical Graphics Technology*, Toronto, August 1983.

[Dowsland93] K. A. Dowsland. Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, 68:389–399, 1993.

[Dowsland95] K. A. Dowsland. Simulated annealing. In C. R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, chapter 2. McGraw-Hill Book Company, Berkshire, 1995.

[Faloutso95] C. Faloutsos and K-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. of Conference on Management of Data (ACM SIGMOD '95)*, pages 163–174, San Jose, CA, May 1995.

[Fienberg79] S. E. Fienberg. Graphical methods in statistics. *The American Statistician*, 33:165–178, 1979.

[Flury81] B. Flury and H. Riedwyl. Graphical representation of multivariate data by means of asymmetrical faces. *Journal of the Ameerican Statistical Association*, 76:757 - 765, 1981.

[Flury97] B. Flury. *A first course in multivariate statistics.* Springer-Verlag, New York, 1997.

[Frick94] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs. In *Proc. of Graph Drawing '94*, volume LNCS 894, pages 388–403, Princeton, New Jersey, October 1994.

[Furnas86] G. W. Furnas. Generalized fisheye views. In *Proc. of Conference on Human Factors in Computing Systems (ACM CHI '86)*, pages 16–23, April 1986.

[Glover95] F. Glover and M. Laguna. Tabu search. In C. R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, chapter 3. McGraw-Hill Book Company, Berkshire, 1995.

[Goldberg89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1989.

[Gordon81] A. D. Gordon. *Classification: Methods for the Exploratory Analysis of Multivariate Data.* Chapman & Hall, London, 1981.

[Gower66] J. C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–338, 1966.

[Gower71] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27:857–871, 1971.

[Gower86] J. C. Gower and P. Legendre. Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3:5–48, 1986.

[Hand94] D. J. Hand et al. *A Handbook of Small Data Sets.* Chapman & Hall, London, 1994.

[Harary69] F. Harary. *Graph Theory.* Addison-Wesley, Reading, Massachusetts, 1969.

[Henderso81] H. V. Henderson and P. F. Velleman. Building regression models interactively. *Biometrics*, 37:391–411, 1981.

[Hotellin33] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology,* 24:417–441, 1933.

[Inselber85] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1:69–91, 1985.

[Joslin99] D. E. Joslin and D. P. Clements. "Squeaky Wheel" optimization. *Journal of Artificial Intelligence Research*, 10:353–373, 1999.

[Kamada89] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, April 1989.

[Keim94] D. A. Keim and H.-P. Kriegel. VisDB: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, 14:40–49, 1994.

[Keselman77] H. J. Keselman and J. C. Rogan. The Tukey multiple comparison test: 1953-1976. *Psychological Bulletin*, 84:1050–1056, 1977.

[Kirkpatr83] S. Kirkpatrik, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[Klock97] H. Klock and J. M. Buhmann. Multidimensional scaling by deterministic annealing. In *Proc. of Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 245–260, Venice, May 1997.

[Kohonen89] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, third edition, 1989.

[Kruskal64] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[Kruskal78a] J. B. Kruskal and J. B. Seery. Designing network diagrams. In *Proc. of the First General Conference on Social Graphics*, pages 22–50, October 1978.

[Kruskal78b] J. B. Kruskal and M. Wish. *Multidimensional Scaling.* Sage Publications, Beverly Hills, California, 1978.

[Lee96] J. P. Lee and G. G. Grinstein. Describing visual interactions to the database: closing the loop between user and data. In *Proc. of Visual Data Exploration and Analysis III*, volume SPIE 2656, pages 93–103, San Jose, California, January 1996.

[Metropol53] N. A. Metropolis et al. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[NIST95] NIST. *Secure Hash Standard - Federal Information Processing Standards Publication 180-1.* U.S. Department of Commerce/National Institute of Standards and Technology, April 1995.

[Ortega70] J. M. Ortega and W. C. Rheinboldt. *Iterative solutions of nonlinear equations in several variables*, pages 253–255. Academic Press, New York, 1970.

[Pearson01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine, Sixth Series*, 2:559–572, 1901.

[Petajan97] E. Petajan, Y. Jean, D. Lieuwen, and V. Anupam. DataSpace: An automated visualization system for large databases. In *Proc. of Visual Data Exploration and Analysis IV*, volume SPIE 3017, pages 89–98, San Jose, California, February 1997.

[Press92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, Cambridge, second edition, 1992.

[Prim57] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, 36:1389–1401, 1957.

[Reeves95] C. R. Reeves. Genetic algorithms. In C. R. Reeves, editor, *Modern Heuristic Techniques for Combinational Problems*, chapter 4. McGraw-Hill Book Company, Berkshire, 1995.

[Rens-Dom95] S. Rens-Domiano and H. E. Hamm. Structural and functional relationships of heterotrimeric G-proteins. *Journal of the Federation of American Societies for Experimental Biology*, 9:1059–1066, 1995.

[Rodden99a] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Evaluating a visualisation of image similarity as a tool for image browsing. In *Proc. of Information Visualization '99*, pages 36–43, San Francisco, October 1999.

[Rodden99b] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Evaluating a visualisation of image similarity (short paper and poster). In *Proc. of Conference on Research and Development in Information Retrieval (ACM SIGIR '99)*, Berkeley, August 1999.

[Rodden00] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. A comparison of measures for visualising image similarity. In *Proc. of Challenge of Image Retrieval*, Brighton, May 2000.

[Rohlf82] F. J. Rohlf. Single-link clustering algorithms. In P. R. Krishnaiah and L. N. Kanal, editors, *Classification Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics*, pages 267–284. North-Holland, 1982.

[Sedgewic92] R. Sedgewick. *Algorithms in C++.* Addison-Wesley, Reading, Massachusetts, 1992.

[Siegel56] S. Siegel. *Nonparametric Statistics for the Behavioral Sciences.* McGraw-Hill Book Company, Berkshire, 1956.

[Spence74] I. Spence and D. W. Domoney. Single subject incomplete designs for nonmetric multidimensional scaling. *Psychometrika*, 39:469–490, 1974.

[Surry95] P. D. Surry. *Hypertext Manual and Glossary of Evolutionary Computing.* http://tarantula.quadstone.co.uk/rpl2/glossary/node1.html, July 1995.

[Szu87] H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122:157–162, June 1987.

[Takane77] Y. Takane, F. W. Young, and J. de Leeuw. Nonmetric individual differences multidimensional scaling: An alternating least-squares method with optimal scaling features. *Psychometrika*, 42:7–67, 1977.

[Tout97] C. A. Tout, S. J. Aarseth, O. R. Pols, and P. P. Eggleton. Rapid binary star evolution for N-body simulations and population synthesis. *Monthly Notices of the Royal Astronomical Society*, 291:732–748, November 1997.

[Tucker51] L. R. Tucker. A method for the synthesis of factor analysis studies. Technical Report 984, Department of the Army, Washington, DC, 1951.

[Weeks79] D. G. Weeks and P. M. Bentler. A comparison of linear and monotone multidimensional scaling models. *Psychological Bulletin*, 86:349–354, 1979.

[Wright89] M. B. Wright. Applying stochastic algorithms to a locomotive scheduling problem. *Journal of Operational Research Society*, 40:187–192, 1989.

[Young70] F. W. Young. Nonmetric multidimensional scaling: Recovery of metric information. *Psychometrika*, 35:455–473, 1970.

[Young72] F. W. Young and N. Cliff. Interactive scaling with individual subjects. *Psychometrika*, 37:385–415, 1972.

Author Index

Subject Index