THE MAJORIZATION APPROACH TO MULTIDIMENSIONAL SCALING

SOME PROBLEMS AND EXTENSIONS

Patrick J.F. Groenen

DSWO PRESS

THE MAJORIZATION APPROACH TO MULTIDIMENSIONAL SCALING

SOME PROBLEMS AND EXTENSIONS

Patrick J.F. Groenen

DEPARTMENT OF DATA THEORY,
Faculty of Social and Behavioural Sciences

1993 DSWO Press, Leiden University, The Netherlands

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Groenen, Patrick J.F.

The majorization approach to multidimensional scaling: some problems and extensions / Patrick J.F. Groenen. — Leiden: DSWO Press, Leiden University. — Ill. — (M&T series, ISSN 0928-1088; 26)
Also publ. as thesis Leiden, 1993. — With ref.
ISBN 90-6695-086-2
Subject headings: multidimensional scaling / majorization / global optimization.

© Cover representation: 'Tunnel'
© Cover design: Handan Arık
Printed by 'Reprodienst Faculteit Sociale Wetenschappen, Rijksuniversiteit Leiden' and by
Printing office 'Karstens drukkers b.v., Leiden'

ISSN 0928-1088-26
ISBN 90-6695-086-2

to Fon and Jeannette

ACKNOWLEDGEMENTS

I would like to thank many persons and friends who have supported me in writing this monograph throughout the last four years. A few people I would like to thank particularly. First, I thank Robert Tijssen for providing me the citation data of journals in earth sciences. Next I owe gratitude to my sister Mirca Groenen who carfully and consistently checked the references and other parts of this monograph. Furthermore, I am very grateful to Handan Arık who produced the cover designs and supported me at times when I needed it most. A special thanks goes to Ivo van der Lans for being a very thorough reader of the first version of the manuscript. I would like to express my gratitude to both John Gower and the referent Rudolf Mathar for many stimulating discussions. Another thanks goes to the people working at the department of Data Theory for providing a pleasant and stimulating environment. I am also indebted to members of the manuscript committee for their remarks. Last but certainly not least, I owe a very special thanks to my promotor Willem Heiser for his support throughout the years and for helping me to make this monograph as it lays in front of you.

CONTENTS

CHAPTER 1

INTRODUCTION AND BASIC INGREDIENTS

This monograph focuses on some technical aspects of multidimensional scaling (MDS). This technique finds a graphical representation of objects in low dimensional space such that the Euclidean distances between objects correspond as well as possible to given dissimilarities that indicate how dissimilar pairs of objects are. The area of multidimensional scaling received a great impetus by the article of Shepard (1962), who gave an heuristic algorithm to solve this problem. The excellent articles of Kruskal (1964a,b) solved the multidimensional scaling problem in a mathematically convenient way. Later several other approaches to perform multidimensional scaling emerged, all based on minimizing a loss function that measures the deviance of the multidimensional scaling model. Loss functions have been defined in a variety of ways, e.g., using the scalar products in classical scaling (Torgerson, 1958; Gower, 1966), using differences between squared distances and squared dissimilarities (the S–STRESS approach of Takane, Young, and De Leeuw, 1977), using the difference of the logarithms of dissimilarities and distances (the maximum likelihood approach of Ramsay, 1977), or using the differences between dissimilarities and distances (Kruskal, 1964a,b). These approaches give solutions which may be different from each other. This monograph discusses some technical problems of Kruskal's approach to multidimensional scaling, and their solutions. Before presenting these problems, we give a brief example of MDS.

Suppose that we have a table of road distances between the capital cities of the 12 members of the EEC as given in Table 1.1. Generally, the entries in such a table are called dissimilarities , since they tell us how dissimilar any pair of objects in the table are. Assume that we do not have a map and only know these dissimilarities. Multidimensional scaling can be used to derive a map in which the distances between the points on the map are as close as possible to the entries in the given table. Figure 1.1 gives such a reconstruction for road distances among capitals in the EEC. We clearly see that the cities are *not* located at their geographical locations. This mismatch arises from the fact that Table 1.1 uses road distances, which are not necessarily equal to the distances "as the crow flies". It is very common in applications of multidimensional scaling that we are not able to find a perfect match between the dissimilarity table and the reconstructed distances in a map. In this example we knew *a priori* that the reconstructed map of the dissimilarity table has to fit reasonably in a two-dimensional plane. For general dissimilarity tables we neither know in advance if a perfect representation exists, nor what the dimensionality should be. Applications of multidimensional scaling can be found many areas of research,

i.e., psychology, sociology, political science, economics, bibliometry, chemical modelling, etcetera. The aim in all applications is to

Table 1.1 *Road distances in kilometres among capitals in the EEC.*

	Ams	Ath	Ber	Bru	Dub	Cop	Lis	Lon	Lux	Mad	Par
Amsterdam											
Athens	3017										
Berlin	660	2552									
Brussels	204	2931	754								
Dublin	1087	3795	1603	921							
Copenhagen	755	3051	445	945	1796						
Lisbon	2301	4559	2887	2092	2821	3045					
London	540	3298	1093	378	543	1306	2279				
Luxembourg	392	2715	768	217	1135	966	2137	592			
Madrid	1765	3945	2350	1559	2247	2495	653	1721	1602		
Paris	494	3025	1087	296	997	1249	1798	455	338	1263	
Rome	1718	2450	1520	1527	2413	2046	2720	1870	1312	2084	1441

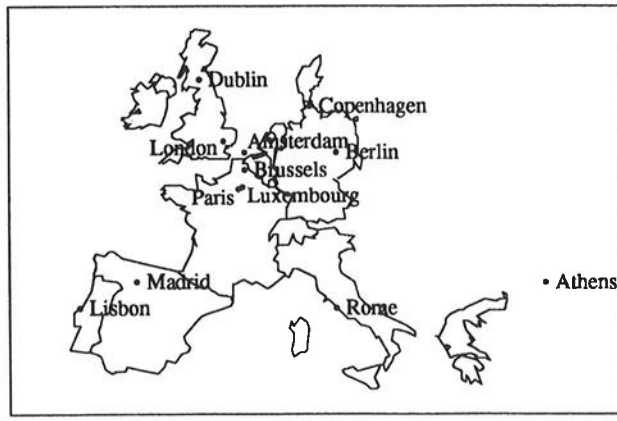


Figure 1.1 *An example of MDS. Map reconstructed from the road distances between the capitals of the EEC reported in Table 1.1. A true map of EEC countries is projected onto the solution.*

find a mapping of objects in a low dimensional space (usually a one, two, or three dimensional space) such that the distances between the objects in the reconstructed map match the given dissimilarities as closely as possible. The graphical representation of the objects can help reveal underlying mechanisms causing the (dis)similarity between the

objects. The dissimilarities need not necessarily stem from real distances, as was the case in the example. They could be obtained in a psychological judgement experiment in which respondents have to compare objects directly. Or, dissimilarities may have been derived from similarity measures (for a list of some of these measures, see, for example, Gower and Legendre, 1986). For a general introduction to multidimensional scaling we refer the reader to Kruskal and Wish (1978), Borg and Lingoes (1987), Schiffman, Reynolds, and Young (1981), or Coxon (1982).

We now specify multidimensional scaling (MDS) more formally. The central MDS loss function in this monograph is the STRESS function that is defined as

$$\sigma^2(\mathbf{X}) = \sum_{i<j} w_{ij}\left(\delta_{ij} - d_{ij}(\mathbf{X})\right)^2. \tag{1.1}$$

In (1.1) the nonnegative dissimilarity between objects i and j is presented by δ_{ij}, their reconstructed Euclidean distance in the plot by $d_{ij}(\mathbf{X})$, the p coordinates of all n objects are gathered in the $n \times p$ matrix $\mathbf{X}$, and fixed nonnegative weights w_{ij} are included to (down)weight the residual of object pair ij. The main algorithm that we use for minimizing STRESS is the majorization algorithm, SMACOF, as proposed by De Leeuw and Heiser (1977, 1980) and De Leeuw (1977, 1988), which can be seen as a generalization of Guttman's (1968) C-matrix method. One of the most attractive features of the SMACOF algorithm is that a series of nonincreasing STRESS values is obtained, and that the difference between configurations in subsequent steps also converges. However, usually the solution at convergence is a local minimum of the STRESS function, not necessarily a global minimum. This local minimum problem is especially severe for unidimensional scaling. Furthermore, convergence of the STRESS values can be very slow due to the linear convergence rate of SMACOF (De Leeuw, 1988). Apart from these computational difficulties, problems can arise with the collection of dissimilarities or with the interpretation of the MDS solution.

In this monograph we propose solutions for some problems with the SMACOF algorithm, without destroying its attractive properties. Majorization is one of the most important concepts that we use in many of the algorithms presented. In particular, an important part of this monograph is devoted to the local minimum problem of STRESS. We investigate several strategies to obtain better local minima, hopefully a global minimum. Furthermore, we look at incomplete MDS, where certain object pairs are disregarded and in which certain objects may have fixed positions in the solution. We present a unification and extension of related algorithms that make use of incomplete MDS. We propose adaptations for certain cases of incomplete MDS that lead to a faster algorithm. A different problem arises when we have to interpret MDS problems with a large number of objects. Then it can be very useful to simplify interpretation by using cluster restrictions. Finally, the SMACOF algorithm is extended to deal with general Minkowski distance, of which the Euclidean distance is a special case. Note that the local minimum problem of MDS remains present when using Minkowski distances. However,

we limit ourselves to the study of the local minimum problem for Euclidean distances, because it turns out to be complicated enough. Often the algorithms and solutions are illustrated with real data, that mostly stem from the social sciences. The interest of this monograph is not confined to MDS only. Many of the technical ideas may prove their usefulness in areas outside the field of MDS.

In the remainder of this chapter we discuss the basic ingredients on which this monograph is built. We present the SMACOF algorithm for MDS and models that can be incorporated as special cases of SMACOF. We also discuss two specific MDS problems, full-dimensional scaling and unidimensional scaling. However, we start by discussing the principle of minimizing a function by iterative majorization, because it is one of the fundamental concepts of this monograph.

1.1 Iterative majorization

In this monograph we often need to find a minimum of a complicated function. One of the main minimization techniques that we shall use is the method of iterative majorization. It is a simple and attractive method that generates a monotonically nonincreasing sequence of function values. If the function is bounded from below we usually end up in a stationary point that is a local minimum. An early reference to majorization in the context of line search can be found in Ortega and Rheinboldt (1970, p. 253-255). Majorization has become increasingly popular as a minimization method (see, for example, Kiers (1990), Bijleveld and De Leeuw (1991), Verboon and Heiser (1992), and Van der Lans (1992)). In the field of multidimensional scaling it has been applied in a variety of settings by, among others, De Leeuw and Heiser (1977), De Leeuw and Heiser (1980), De Leeuw (1988), Meulman (1986, 1992), and Groenen, Mathar, and Heiser (1992).

The central idea of the majorization method is to replace iteratively the original complicated function $\varphi(\mathbf{x})$ by an auxiliary function $\hat{\varphi}(\mathbf{x},\mathbf{y})$, which has to meet the following requirements. First, the auxiliary function $\hat{\varphi}(\mathbf{x},\mathbf{y})$ should be more simple to minimize than $\varphi(\mathbf{x})$. Secondly, the original function must always be smaller than or at most equal to the auxiliary function, i.e., $\varphi(\mathbf{x}) \leq \hat{\varphi}(\mathbf{x},\mathbf{y})$. Thirdly, the auxiliary function should touch the surface at the so-called supporting point $\mathbf{y}$, i.e., $\varphi(\mathbf{y}) = \hat{\varphi}(\mathbf{y},\mathbf{y})$. If these three requirements are met, we call $\hat{\varphi}(\mathbf{x},\mathbf{y})$ a majorizing function of $\varphi(\mathbf{x})$.

To understand the principle of minimizing a function by majorization, consider the following. Let the minimum of $\hat{\varphi}(\mathbf{x},\mathbf{y})$ over $\mathbf{x}$ be attained at $\mathbf{x}^*$, for $\mathbf{x}$, $\mathbf{y}$, $\mathbf{x}^*$ in the corresponding domain X. The requirements of the majorizing function imply the chain of inequalities

$$\varphi(\mathbf{x}^*) \leq \hat{\varphi}(\mathbf{x}^*,\mathbf{y}) \leq \hat{\varphi}(\mathbf{y},\mathbf{y}) = \varphi(\mathbf{y}) \tag{1.2}$$

for all $\mathbf{x}, \mathbf{y} \in X$. This chain of inequalities is named the *sandwich* inequality by De Leeuw (1992), since the minimum of the majorizing function $\hat{\varphi}(\mathbf{x}^*,\mathbf{y})$ is squeezed between $\varphi(\mathbf{x}^*)$

and $\varphi(y)$. A graphical representation of these inequalities is presented in Figure 1.2 for two subsequent iterations of iterative majorization of the function $\varphi(x)$, with x being a scalar.

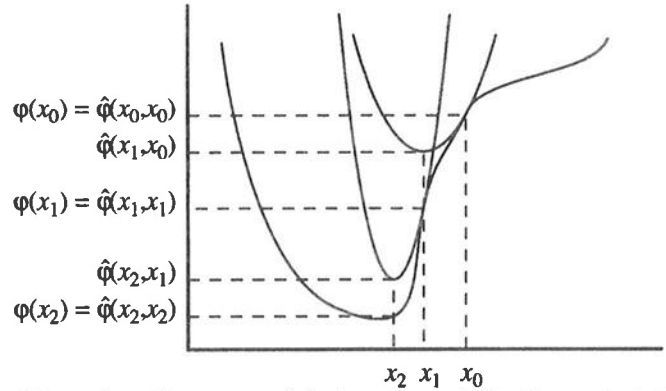


Figure 1.2 *Illustration of two steps of the iterative majorization method. The auxiliary function $\hat{\varphi}(x,x_0)$ is located above the original function $\varphi(x)$ and touches at the supporting point x_0. The minimum of the auxiliary function $\hat{\varphi}(x,x_0)$ is attained at x_1, where $\varphi(x_1)$ can never be larger than $\hat{\varphi}(x_1,x_0)$. These steps are repeated.*

The majorization algorithm is given by

1. $y \leftarrow y_0$, where y_0 is a starting value.
2. Find x^+ for which $\hat{\varphi}(x^+,y) = \min_x \hat{\varphi}(x,y)$.
3. If $\varphi(y) - \varphi(x^+) < \varepsilon$ then stop. (ε a small positive constant.)
4. $y \leftarrow x^+$ and go to 2.

Obviously, by (1.2) the majorization algorithm yields a nonincreasing sequence of function values, which is an attractive aspect of iterative majorization. If the function $\varphi(x)$ is not bounded from below and if there are no sufficient restrictions on x, then the stop criterion of step 3 may never be met. In the sequel, this situation does not arise. Although the function value never increases, the majorization principle does not say how fast the function values converge. In the next section, we obtain a convergence theorem for a specific case of majorization. A more relaxed version of the majorization algorithm is obtained by demanding in step 2 merely that $\hat{\varphi}(x^*,y) \leq \hat{\varphi}(y,y)$, instead of requiring that x^* is the minimum of $\hat{\varphi}(x,y)$. This weaker form of majorization does not change the sandwich inequality (1.2), so that a reduction of function values is retained. Generally, we do not know much about the convergence of the sequence of x, except if we have additional properties of the corresponding iterative map such as closeness and continuity (see Zangwill, 1969). A necessary condition for a point x^* to be a minimizer of $\varphi(x)$ is

that $\mathbf{x}^*$ minimizes $\hat{\varphi}(.,\mathbf{x}^*)$. So, if $\varphi(\mathbf{x}^+) = \varphi(\mathbf{y})$ and $\mathbf{x}^+ = \mathbf{y}$, this necessary condition is satisfied by $\mathbf{y}$. Note that the majorization algorithm can stop at a stationary point that is not a local minimum. However, Fletcher (1987, p. 19) notes that for algorithms that reduce the function value on every iteration, it usually holds that "the stationary point turns out to be a local minimizer, except in rather rare circumstances".

We distinguish two particularly useful applications of majorization. The first one is majorization of a concave function, which we call *linear* majorization. Any concave function $\varphi(\mathbf{x})$ may by definition be majorized by a linear function at any point $\mathbf{x}$. Thus, we can always find a plane that touches at the supporting point $\mathbf{y}$. This plane may touch the concave function elsewhere, or is located above it. Moreover, the plane is a simple linear function of $\mathbf{x}$. An example of a univariate concave function is given in Figure 1.3.

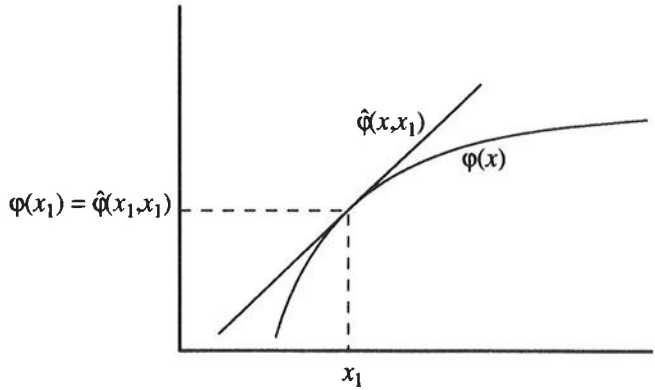


Figure 1.3 *An example of linear majorization of a concave function $\varphi(x)$ by a linear function.*

To see why linear majorization holds for concave functions, it is convenient to apply the concept of subgradient from convex analysis (see, e.g., Rockafellar, 1970). The set of all subgradients of $\varphi(.)$ at $\mathbf{y}$ is called subdifferential $\partial\varphi(\mathbf{y})$. If $\varphi(.)$ is differentiable at $\mathbf{y}$ then the set of subgradients consists of one element, which is the gradient. Thus, if a function is differentiable then the derivative is the only subgradient. Let $\mathbf{b}(\mathbf{y})$ be a subgradient at $\mathbf{y}$ of the concave function $\varphi(\mathbf{y}) = \mathbf{y}'\mathbf{b}(\mathbf{y}) + c(\mathbf{y})$. For $\mathbf{b}(\mathbf{y})$ to be a subgradient, it must satisfy the subgradient inequality

$$\varphi(\mathbf{x}) \leq \varphi(\mathbf{y}) + (\mathbf{x} - \mathbf{y})'\mathbf{b}(\mathbf{y}) = \mathbf{x}'\mathbf{b}(\mathbf{y}) + c(\mathbf{y}) = \hat{\varphi}(\mathbf{x},\mathbf{y}). \tag{1.3}$$

The inequality shows that the majorization requirements $\varphi(\mathbf{x}) \leq \hat{\varphi}(\mathbf{x},\mathbf{y})$ and $\varphi(\mathbf{x}) = \hat{\varphi}(\mathbf{x},\mathbf{x})$ are fulfilled. Moreover, the majorizing function $\hat{\varphi}(\mathbf{x},\mathbf{y}) = \mathbf{x}'\mathbf{b}(\mathbf{y}) + c(\mathbf{y})$ is a linear function in $\mathbf{x}$, which is clearly a simple function of $\mathbf{x}$. As an example, we can apply linear majorization to the concave function $\varphi(x) = \log x$ for $x > 0$. The (sub)gradient of $\log y$ is

given by $1/y$, so that the subgradient inequality yields $\varphi(x) = \log x \leq \log y + (x - y)/y = \hat{\varphi}(x,y)$. It is not difficult to see that choosing $x = y$ gives equality.

The second group of functions that can be majorized is characterized by a bounded Hessian. This type of majorization can be applied if the function $\varphi(\mathbf{x})$ can be majorized by $\hat{\varphi}(\mathbf{x},\mathbf{y}) = \mathbf{x}'\mathbf{A}(\mathbf{y})\mathbf{x} - \mathbf{x}'\mathbf{b}(\mathbf{y}) + c(\mathbf{y})$, with $\mathbf{A}(\mathbf{y})$ positive semi-definite. A bounded Hessian $\nabla^2\varphi(\mathbf{x})$ implies that the curvature (that is, the second derivative) of the function $\varphi(\mathbf{x})$ is always less than the curvature of some quadratic function, $\mathbf{x}'\nabla^2\varphi(\mathbf{x})\mathbf{x} \leq \mathbf{x}'\mathbf{A}(\mathbf{y})\mathbf{x}$. Therefore, we call this type of majorization *quadratic* majorization. A nice example is the function $\varphi(x) = |x|$, used by Heiser (1988). It can be proved that the function $\hat{\varphi}(x,y) = \frac{1}{2}|y| + \frac{1}{2}x^2/|y|$ for $y \neq 0$ is a majorizing function of $\varphi(x)$. This example of quadratic majorization is illustrated in Figure 1.4.

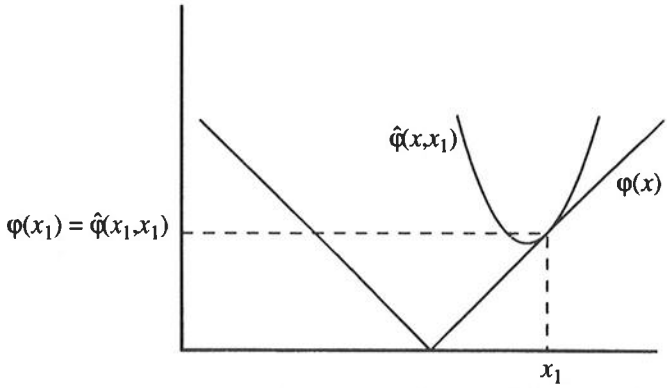


Figure 1.4 *An example of quadratic majorization of* $\varphi(x) = |x|$, *a function with a bounded second derivative if* $x \neq 0$.

A special case of quadratic majorization evolves by majorizing a function that is a sum of a quadratic function in a fixed metric $\mathbf{A}$ ($\mathbf{A}$ is independent of $\mathbf{y}$) and a concave function. Later we shall see that the STRESS function is such a special case of quadratic majorization.

The classification of linear and quadratic majorization is due to De Leeuw (1992), who used the terms type I and type II majorization. When minimizing a function we may apply linear and quadratic majorization together without any problem, as long as the majorization conditions hold.

1.2 Convergence rate of quadratic majorization

Throughout this monograph the algorithms based on majorization play an important role. Therefore, it is useful to be able to have an expected rate of convergence of such

algorithms. In particular, we wish to establish how fast $\mathbf{x}$ converges to a stationary value $\mathbf{x}^*$. Assessing the convergence behaviour of $\mathbf{x}$ is done by examining the ratio

$$\lim_{k\to\infty} \frac{\left\|\mathbf{x}_{k+1} - \mathbf{x}^*\right\|}{\left\|\mathbf{x}_k - \mathbf{x}^*\right\|^q} = \kappa, \tag{1.4}$$

where q is the *order* of convergence and κ is the *rate* of convergence. For $q = 1$ and $\kappa \geq 1$ we have *sublinear* divergence, *linear* convergence for $0 < \kappa < 1$, and for $\kappa = 0$ *superlinear* convergence. If $q = 2$ and $0 < \kappa < 1$ we have *quadratic* convergence, and for $\kappa = 0$ *superquadratic* convergence. For the class of quadratic majorizing functions, the next theorem tells that the majorization algorithm has linear convergence rate if the function is twice differentiable at the local minimum and the largest eigenvalue of the derivative of the iterative map is smaller than 1.

Theorem 1.1
Suppose that the function $\varphi(\mathbf{x})$ is majorized by $\hat{\varphi}(\mathbf{x},\mathbf{y}) = \frac{1}{2}\mathbf{x}'\mathbf{A}(\mathbf{y})\mathbf{x} - \mathbf{x}'\mathbf{b}(\mathbf{y}) + c(\mathbf{y})$, with equality if $\mathbf{y} = \mathbf{x}$, i.e., $\varphi(\mathbf{x}) \leq \hat{\varphi}(\mathbf{x},\mathbf{y})$ and $\varphi(\mathbf{x}) = \hat{\varphi}(\mathbf{x},\mathbf{x})$. Let one iteration of the majorization algorithm be given by

$$\begin{aligned} \mathbf{x}^+ &\leftarrow \mathbf{A}(\mathbf{y})^{-1}\mathbf{b}(\mathbf{y}) \\ \mathbf{y} &\leftarrow \mathbf{x}^+, \end{aligned} \tag{1.5}$$

where we assume that the inverse $\mathbf{A}(\mathbf{y})^{-1}$ exists. Also, assume that $\mathbf{x}^*$ is a local minimum of $\varphi(\mathbf{x})$ and that $\mathbf{x}^+$ converges to $\mathbf{x}^*$. Furthermore, assume that the Hessian of $\mathbf{y}'\mathbf{b}(\mathbf{y})$ exists at $\mathbf{x}^*$, and is given by $\mathbf{B}(\mathbf{x}^*)$. Then the quadratic majorization algorithm has linear convergence with rate equal to the largest eigenvalue λ of $\mathbf{A}(\mathbf{x}^*)^{-1}\mathbf{B}(\mathbf{x}^*)$, if $0 < \lambda < 1$.

Proof
By lemma 10.2.1 of Ortega and Rheinboldt (1970), the derivative of the mapping

$$\mathbf{x}^+ \leftarrow \mathbf{A}(\mathbf{x})^{-1}\mathbf{b}(\mathbf{x}) \tag{1.6}$$

at the local minimum $\mathbf{x}^*$ is given by

$$\mathbf{A}(\mathbf{x}^*)^{-1}\mathbf{B}(\mathbf{x}^*), \tag{1.7}$$

because the derivative of the mapping $\mathbf{x}^{++} \leftarrow \mathbf{x} - \mathbf{A}(\mathbf{x})^{-1}\mathbf{b}(\mathbf{x})$ can be expressed by $\mathbf{I} - \mathbf{A}(\mathbf{x}^*)^{-1}\mathbf{B}(\mathbf{x}^*)$. Next, we need to prove that the eigenvalues of $\mathbf{A}(\mathbf{x}^*)^{-1}\mathbf{B}(\mathbf{x}^*)$ are real. Note that because of the majorization inequality and the assumption that $\mathbf{A}(\mathbf{y})$ is invertible, $\mathbf{A}(\mathbf{y})$ is symmetric and positive definite and so is $\mathbf{A}(\mathbf{y})^{-1}$. Furthermore, $\mathbf{B}(\mathbf{y})$ is the Hessian of $\mathbf{y}'\mathbf{b}(\mathbf{y})$ and is consequently symmetric. Wilkinson (1965, page 35)

proved that for $\mathbf{A}(\mathbf{x}^*)^{-1}$ symmetric and positive definite and for $\mathbf{B}(\mathbf{x}^*)$ symmetric, the eigenvalues of $\mathbf{A}(\mathbf{x}^*)^{-1}\mathbf{B}(\mathbf{x}^*)$ are real, because the eigensystem $(\mathbf{A}(\mathbf{x}^*)^{-1}\mathbf{B}(\mathbf{x}^*) - \lambda\mathbf{I})\mathbf{y} = \mathbf{0}$ is equivalent to $(\mathbf{B}(\mathbf{x}^*) - \lambda\mathbf{A}(\mathbf{x}^*))\mathbf{y} = \mathbf{0}$.
The theorems of Ostrowski (1966, page 162, theorem 22.1) and Ortega and Rheinboldt, 1970, theorem 10.1.4) say that if the derivative of a mapping has the largest eigenvalue $\lambda < 1$ at a local minimum $\mathbf{x}^*$, then the iterative process converges with a linear convergence rate of λ to $\mathbf{x}^*$. Q.E.D.

This theorem has implications for several applications of majorization. It allows one, e.g., to prove the linear convergence rate of STRESS under certain conditions. For many applications of majorization, this result can be applied to prove linear convergence.

1.3 Majorizing the STRESS function

In this section we apply iterative majorization to the STRESS function, which goes back to De Leeuw and Heiser (1977), De Leeuw and Heiser (1980), and De Leeuw (1988). The acronym SMACOF initially stood for Scaling by MAximizing a COnvex Function, but since the mid 1980's it stands for Scaling by MAjorizing a COmplicated Function. Algorithms other than SMACOF have been derived to minimize STRESS. For example, using approaches from convex analysis, very similar algorithms for minimizing STRESS were obtained by De Leeuw (1977), Mathar (1989), and Mathar and Groenen (1991). Here, we pursue the majorization approach and show how to majorize the STRESS function $\sigma(\mathbf{X})$, following the SMACOF theory.

The STRESS function (1.1) can be rewritten as

$$\begin{aligned}\sigma^2(\mathbf{X}) &= \sum_{i<j} w_{ij}\left(\delta_{ij} - d_{ij}(\mathbf{X})\right)^2 \\ &= \sum_{i<j} w_{ij}\delta_{ij}^2 + \sum_{i<j} w_{ij}\, d_{ij}^2(\mathbf{X}) - 2\sum_{i<j} w_{ij}\delta_{ij}\, d_{ij}(\mathbf{X}) \\ &= \eta_\delta^2 + \eta^2(\mathbf{X}) - 2\rho(\mathbf{X}),\end{aligned} \tag{1.8}$$

where $d_{ij}(\mathbf{X})$ is the Euclidean distance,

$$d_{ij}(\mathbf{X}) = \left(\sum_{s=1}^{p}\left(x_{is} - x_{js}\right)^2\right)^{1/2} = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|, \tag{1.9}$$

between object i and j, and the row vector $\mathbf{x}_i$ contains the coordinates of object i. We assume throughout this monograph that the weight matrix is irreducible, i.e., there exists no partitioning of objects into disjoint subsets, such that $w_{ij} = 0$ whenever objects i and j are in different subsets. If the weight matrix is reducible, then the problem can be decomposed in separate smaller multidimensional scaling problems, one for each subset.

In the sequel, we assume without loss of generality that $\mathbf{X}$ has zero column means. By using the Cauchy-Schwarz inequality

$$\|\mathbf{x}_i - \mathbf{x}_j\|\|\mathbf{y}_i - \mathbf{y}_j\| \geq (\mathbf{x}_i - \mathbf{x}_j)'(\mathbf{y}_i - \mathbf{y}_j), \tag{1.10}$$

it can be verified that

$$-w_{ij}\delta_{ij}\, d_{ij}(\mathbf{X}) \leq -\frac{w_{ij}\delta_{ij}}{d_{ij}(\mathbf{Y})}(\mathbf{x}_i - \mathbf{x}_j)'(\mathbf{y}_i - \mathbf{y}_j) \tag{1.11}$$

holds whenever $d_{ij}(\mathbf{Y}) \neq 0$. Equality is obtained if $\mathbf{X}$ equals $c\mathbf{Y}$. Furthermore, if $d_{ij}(\mathbf{Y}) = 0$ it remains true that

$$-w_{ij}\delta_{ij}\, d_{ij}(\mathbf{X}) \leq 0 \tag{1.12}$$

with equality if $d_{ij}(\mathbf{X}) = d_{ij}(\mathbf{Y}) = 0$. Thus, for $b_{ij}(\mathbf{Y}) = w_{ij}\delta_{ij}/d_{ij}(\mathbf{Y})$ if $d_{ij}(\mathbf{Y}) \neq 0$ and $b_{ij}(\mathbf{Y}) = 0$ if $d_{ij}(\mathbf{Y}) = 0$ it always holds that

$$-w_{ij}\delta_{ij}d_{ij}(\mathbf{X}) \leq -b_{ij}(\mathbf{Y})\,(\mathbf{x}_i - \mathbf{x}_j)'(\mathbf{y}_i - \mathbf{y}_j), \tag{1.13}$$

with equality if $\mathbf{X} = c\mathbf{Y}$. Summing both sides of (1.13) over $i < j$ shows that

$$-\rho(\mathbf{X}) = -\sum_{i<j} w_{ij}\delta_{ij}d_{ij}(\mathbf{X}) \leq -\sum_{i<j} b_{ij}(\mathbf{Y})\,(\mathbf{x}_i - \mathbf{x}_j)'(\mathbf{y}_i - \mathbf{y}_j) = -\hat{\rho}(\mathbf{X},\mathbf{Y}). \tag{1.14}$$

This implies that $-\rho(\mathbf{X})$ can be majorized by a linear function $-\hat{\rho}(\mathbf{X},\mathbf{Y})$ so that $\rho(\mathbf{X}) = \hat{\rho}(\mathbf{X},\mathbf{X})$ and $-\rho(\mathbf{X}) \leq -\hat{\rho}(\mathbf{X},\mathbf{Y})$. This is a clear example of linear majorization.

We shall rewrite $\rho(\mathbf{X})$ and $\hat{\rho}(\mathbf{X},\mathbf{Y})$ in such a way that a matrix expression is obtained from which an update sequence for the coordinates of the objects is easily derived. An important aid is the expression

$$\sum_{i<j} a_{ij}(\mathbf{x}_i - \mathbf{x}_j)'(\mathbf{y}_i - \mathbf{y}_j) = \operatorname{tr}\mathbf{X}'\mathbf{A}\mathbf{Y}, \tag{1.15}$$

which holds for every symmetric matrix $\mathbf{A}$ with off-diagonal elements $-a_{ij}$ and the diagonal elements $\sum_{j\neq i} a_{ij}$. The rows and columns of $\mathbf{A}$ sum to zero, which implies that $\mathbf{1}$ is contained in the null space of $\mathbf{A}$, so that the rank of $\mathbf{A}$ can never exceed $n - 1$. If all elements a_{ij} are nonnegative then $\mathbf{A}$ is positive semi-definite. Combining (1.15) and the right side of (1.14) allows us to express the majorizing function $\hat{\rho}(\mathbf{X},\mathbf{Y})$ as

$$\operatorname{tr}\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y}. \tag{1.16}$$

Since equality of (1.14) occurs for $\mathbf{Y} = \mathbf{X}$, we have the majorizing expression

$$-\rho(\mathbf{X}) = -\hat{\rho}(\mathbf{X},\mathbf{X}) = -\text{tr } \mathbf{X}'\mathbf{B}(\mathbf{X})\mathbf{X} \leq -\text{tr } \mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y} = -\hat{\rho}(\mathbf{X},\mathbf{Y}). \tag{1.17}$$

Similarly, we may use (1.15) to obtain a matrix expression for $\eta^2(\mathbf{X})$. Let $\mathbf{D}^{(2)}$ be the matrix of squared distances $d_{ij}^2(\mathbf{X})$, i.e.,

$$\mathbf{D}^{(2)} = \mathbf{1}\alpha' + \alpha\mathbf{1}' - 2\mathbf{X}\mathbf{X}', \tag{1.18}$$

where α is a vector with the diagonal elements of $\mathbf{X}\mathbf{X}'$. Since $\mathbf{D}^{(2)}$ has zero diagonal, $\eta^2(\mathbf{X})$ can also be written as

$$-\tfrac{1}{2}\,\text{tr } \mathbf{V}\mathbf{D}^{(2)}, \tag{1.19}$$

where $\mathbf{V}$ is a symmetric matrix with off-diagonal elements $-w_{ij}$ and diagonal elements $\sum_{j\neq i} w_{ij}$. The factor 1/2 is introduced because of the symmetry of $\mathbf{D}$. Manipulating (1.18) and (1.19) and using the fact that $\mathbf{1}$ is in the null-space of $\mathbf{V}$ gives

$$\eta^2(\mathbf{X}) = \text{tr } \mathbf{X}'\mathbf{V}\mathbf{X} \ . \tag{1.20}$$

Formulations (1.20) and (1.17) allow us to write the STRESS function conveniently in matrix algebra by

$$\sigma^2(\mathbf{X}) = \eta_\delta^2 + \text{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\text{tr}\mathbf{X}'\mathbf{B}(\mathbf{X})\mathbf{X} \ , \tag{1.21}$$

which is majorized by

$$\hat{\sigma}^2(\mathbf{X},\mathbf{Y}) = \eta_\delta^2 + \text{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\text{tr}\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y} \ . \tag{1.22}$$

The majorizing function (1.22) is a quadratic function in $\mathbf{X}$. Its minimum is obtained by setting its gradient with respect to $\mathbf{X}$ equal to zero;

$$\nabla\,\hat{\sigma}^2(\mathbf{X},\mathbf{Y}) = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{Y})\mathbf{Y} = \mathbf{0} \ . \tag{1.23}$$

As stated above, the matrix $\mathbf{V}$ has $\mathbf{1}$ in its null space. Therefore, we need a generalized inverse of $\mathbf{V}$, for which we take the Moore-Penrose inverse, to obtain the minimum of $\hat{\sigma}^2(\mathbf{X},\mathbf{Y})$. The Moore-Penrose inverse of $\mathbf{V}$ is given by $\mathbf{V}^- = (\mathbf{V} + \mathbf{1}\mathbf{1}')^{-1} - n^{-2}\mathbf{1}\mathbf{1}'$. The last term $-n^{-2}\mathbf{1}\mathbf{1}'$ is irrelevant in SMACOF as $\mathbf{V}^-$ is subsequently multiplied by a matrix orthogonal to $\mathbf{1}$, since $\mathbf{B}(\mathbf{Y})$ also has $\mathbf{1}$ in its null space . This leads us to the update formula of the SMACOF algorithm,

$$\mathbf{X} = \mathbf{V}^-\mathbf{B}(\mathbf{Y})\mathbf{Y}. \tag{1.24}$$

De Leeuw and Heiser (1980) call (1.24) the Guttman transform, in recognition of Guttman (1968). The majorization algorithm guarantees a series of nonincreasing STRESS

values. When the algorithm stops, the stationary condition $\mathbf{X} = \mathbf{V}^{-}\mathbf{B}(\mathbf{X})\mathbf{X}$ holds. Note that after one step of the algorithm $\mathbf{X}$ is column-centered, even if $\mathbf{Y}$ is not column-centered. Since distances do not change under rotation of $\mathbf{X}$, it is convenient to rotate $\mathbf{X}$ to principal axes after convergence has been reached.

In the next few sections we look at two special cases of MDS, i.e., unidimensional scaling and full-dimensional scaling. However, first we discuss a coordinate-free formulation of the SMACOF algorithm, and the rate of convergence of the Guttman transform.

1.3.1 Coordinate-free update

Using the Guttman transform (1.24) allows us to develop an update sequence of SMACOF in terms of distances only (Gower, 1991). One of the advantages is that it eliminates the rotation and translation problem. It also has some elegance to formulate the algorithm of a distance model like STRESS in terms of distances only.

From (1.18) it is known that the inner product matrix $-2\mathbf{XX}'$ is equal to the double centered distance matrix $\mathbf{J}\mathbf{D}^{(2)}\mathbf{J}$, where $\mathbf{D}^{(2)}$ contains squared distances and $\mathbf{J}$ is the centering operator $\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}'$. This suggests immediately post-multiplying the Guttman transform with its transpose, i.e.,

$$\mathbf{X}\mathbf{X}' = \mathbf{V}^{-}\mathbf{B}(\mathbf{Y})\mathbf{Y}\mathbf{Y}'\mathbf{B}(\mathbf{Y})\mathbf{V}^{-}, \tag{1.25}$$

which essentially gives the coordinate free update. Furthermore, we may replace $\mathbf{Y}\mathbf{Y}'$ by $\mathbf{D}_{\mathbf{Y}}^{(2)}$ since it is pre and post multiplied by $\mathbf{B}(\mathbf{Y})$, which is rowwise and columnwise orthogonal to $\mathbf{1}$. To express that the elements of $\mathbf{B}(\mathbf{Y})$ only depend on the $d_{ij}(\mathbf{Y})$, not on $\mathbf{Y}$ itself, we write it as $\mathbf{B}(\mathbf{D}_{\mathbf{Y}})$. This leads to the following distance formulation of the Guttman transform:

$$\mathbf{J}\mathbf{D}^{(2)}\mathbf{J} = \mathbf{V}^{-}\mathbf{B}(\mathbf{D}_{\mathbf{Y}})\mathbf{D}_{\mathbf{Y}}^{(2)}\mathbf{B}(\mathbf{D}_{\mathbf{Y}})\mathbf{V}^{-}. \tag{1.26}$$

Note in this coordinate-free formulation the rank of the starting configuration is retained throughout the iterative process, even without imposing additional rank restrictions. The majorization results still hold, which implies that the sequence of STRESS values never increases. A disadvantage of the coordinate-free update is that it requires more computation and that it may not be numerically stable. Gower (1984) was the first to notice the possibility of coordinate-free scaling by looking at the normal equations at a local minimum. However, his formulation did not include the post multiplication by $\mathbf{B}(\mathbf{D}_{\mathbf{Y}})\mathbf{V}^{-}$ in (1.26) so that the sequence of STRESS values were not necessarily convergent.

1.3.2 Convergence rate of SMACOF

Quite early it was recognized that the SMACOF algorithm has a linear convergence rate (De Leeuw and Heiser, 1980). De Leeuw (1988) proved the linear convergence rate using the fact that if $w_{ij}\delta_{ij} > 0$ for all object pairs ij, no distance can be zero at a local minimum, which makes STRESS twice differentiable at a local minimum (see De Leeuw, 1984, or section 7.4). According to theorem 1.1, we have to find the largest eigenvalue of the derivative of the Guttman transform to establish the convergence rate. Let us first regard the derivative $\mathbf{H}(\mathbf{X})$ of $\mathbf{B}(\mathbf{X})\mathbf{X}$, which is equal to the Hessian of $\rho(\mathbf{X})$. It can be expressed as a partitioned block matrix of $p \times p$ blocks $\mathbf{H}_{st}(\mathbf{X})$ of size $n \times n$. The diagonal blocks $\mathbf{H}_{ss}(\mathbf{X})$ are equal to $(\mathbf{B}(\mathbf{X}) - \mathbf{E}_{ss}(\mathbf{X}))$ and the off diagonal blocks $\mathbf{H}_{st}(\mathbf{X})$ are equal to $-\mathbf{E}_{st}(\mathbf{X})$, where $\mathbf{E}_{st}(\mathbf{X})$ consist of off diagonal elements

$$e_{stij} = -\frac{w_{ij}\delta_{ij}}{d_{ij}^3(\mathbf{X})}(x_{is} - x_{js})(x_{it} - x_{jt}), \tag{1.27}$$

and the diagonal elements $e_{stii} = -\sum_{j \neq i} e_{stij}$. The derivative of the Guttman transform is given by $\mathbf{A}^{-}\mathbf{H}(\mathbf{X})$, where $\mathbf{A}$ is a block diagonal matrix with diagonal blocks $\mathbf{V}$. To establish the convergence rate, we need to know the largest eigenvalue of $\mathbf{A}^{-}\mathbf{H}(\mathbf{X}^*)$, with $\mathbf{X}^*$ a local minimum. Using similar arguments as in theorem 1.1, we can prove that $\mathbf{A}^{-}\mathbf{H}(\mathbf{X}^*)$ has real eigenvalues. At $\mathbf{X}^*$ the Hessian of the STRESS function must be positive semi-definite, i.e., $\mathbf{A} - \mathbf{H}(\mathbf{X}^*)$ must be positive semi-definite. Consequently, $\mathbf{I} - \mathbf{A}^{-}\mathbf{H}(\mathbf{X}^*)$ must be positive semi-definite too, which proves that all eigenvalues of $\mathbf{A}^{-}\mathbf{H}(\mathbf{X}^*)$ are smaller than or equal to 1. De Leeuw (1988) showed that at a local minimum $\mathbf{X}^*$, $\mathbf{A}^{-}\mathbf{H}(\mathbf{X}^*)$ has $1/2p(p-1)$ eigenvalues equal to one, due to the invariance of STRESS under rotations. Let us impose some identification constraints on $\mathbf{X}$, like rotating the solution always to principal axes. Furthermore, suppose that no other eigenvalues of $\mathbf{A}^{-}\mathbf{H}(\mathbf{X}^*)$ are equal to one, so that $\mathbf{X}^*$ is an isolated stationary point, as De Leeuw (1988) calls it. Although the majorizing function (1.22) was obtained by linear majorization, it may be regarded as a quadratic majorizing function with matrix $\mathbf{A}(\mathbf{X})$ being fixed. Then we can use theorem 1.1 which tells us that the SMACOF algorithm defined by (1.24) has a linear convergence rate λ, with λ being the largest eigenvalue of $\mathbf{A}^{-}\mathbf{H}(\mathbf{X}^*)$ smaller than 1. For more details, we refer to De Leeuw (1988).

1.4 Full-dimensional scaling

A special case of minimizing STRESS appears in full-dimensional scaling, where the dimensionality is at most $p = n - 1$. In full-dimensional scaling there is only one minimum that is a global one, which is a result due to De Leeuw (1992). This can be seen from the matrix of squared distances $\mathbf{D}^{(2)}$ which equals $\mathbf{1}\boldsymbol{\alpha}' + \boldsymbol{\alpha}\mathbf{1}' - 2\mathbf{X}\mathbf{X}'$, with $\mathbf{X}$ being column-centered, cf. (1.18). Thus the rank of $\mathbf{X}\mathbf{X}'$ can never exceed $n - 1$. For $p = n - 1$

the cross product term $\mathbf{XX}'$ is simply a double centered positive semi definite matrix $\mathbf{K}$, so that the squared distances $d_{ij}^2(\mathbf{X})$ are given by $k_{ii} + k_{jj} - 2k_{ij}$. This allows us to express STRESS as

$$\begin{aligned}\sigma^2(\mathbf{K}) &= \eta_\delta^2 + \sum_{i<j} w_{ij}(k_{ii} + k_{jj} - 2k_{ij}) - 2\sum_{i<j} w_{ij}\delta_{ij}\sqrt{k_{ii} + k_{jj} - 2k_{ij}} \\ &= \eta_\delta^2 + \eta^2(\mathbf{K}) - 2\rho(\mathbf{K}).\end{aligned} \tag{1.28}$$

The term $\sum_{i<j} w_{ij}(k_{ii} + k_{jj} - 2k_{ij})$ is a linear function in $\mathbf{K}$. The second term takes minus the square root of the same linear function of $\mathbf{K}$, which is a convex function in $\mathbf{K}$. The sum of a linear and a convex function is convex. Thus minimizing STRESS over $\mathbf{K}$ is minimizing a convex function over a convex set, which has a local minimum that is a global minimum. Note that this result does not hold in case $\mathbf{K}$ is restricted to have $p < n - 1$, because the set of $\mathbf{K}$ restricted to have rank $p < n - 1$ is not convex. A similar result holds for full-dimensional scaling with S–STRESS, which among others has been proved by Gaffke and Mathar (1989) using the cyclic projection algorithm.

Although one would expect $\mathbf{K}$ to be of rank $n - 1$ at a minimum, this usually is not the case. In fact, numerical experiments suggest that at the minimum, the rank of $\mathbf{K}$ does not exceed the number of positive eigenvalues in classical scaling. Bailey and Gower (1990) proved this conjecture for S–STRESS, but we were not able to prove it for STRESS.

1.5 Unidimensional scaling

It has been noted by Heiser and De Leeuw (1977), Defays (1978), and Hubert and Arabie (1986) that minimizing the STRESS function with equal weights changes to a combinatorial problem when $p = 1$. In this section we show why unidimensional scaling is a combinatorial problem and extend this result to unequal weights.

If we are dealing with one dimension only, the distance between two points can be expressed as $d_{ij}(\mathbf{x}) = (x_i - x_j)\text{sign}(x_i - x_j)$, where $\text{sign}(x_i - x_j) = 1$ for $x_i > x_j$, $\text{sign}(x_i - x_j) = 0$ for $x_i = x_j$ and $\text{sign}(x_i - x_j) = -1$ for $x_i < x_j$. Here we use $\mathbf{x}$ for the column vector of coordinates of the n objects. Thus, we see that only the rankorder of $\mathbf{x}$ determines the $\text{sign}(x_i - x_j)$. In this case, STRESS can be expressed as

$$\sigma^2(\mathbf{x}) = \eta_\delta^2 + \mathbf{x}'\mathbf{V}\mathbf{x} - 2\sum_{i<j} w_{ij}\delta_{ij}(x_i - x_j)\text{sign}(x_i - x_j). \tag{1.29}$$

This shows that the cross-product term of STRESS, $\rho(\mathbf{x})$, can be factored into a part that is linear in $\mathbf{x}$ and a part that depends only on the rankorder of the elements of $\mathbf{x}$. As an example we plotted in Figure 1.5 the values of $\rho(\mathbf{x})$ and the values of the gradient

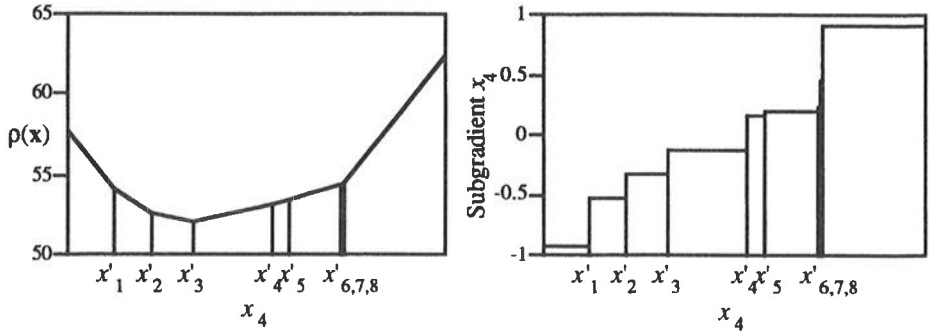


Figure 1.5 $\rho(\mathbf{x})$ *and its subgradient values for varying* x_4 *keeping the other coordinates fixed at their optimal position* x'_i*. The dissimilarities are taken from the Mani-collection from Robinson (1951) reported by Hubert and Arabie (1986).*

of $\rho(\mathbf{x})$ against varying values of x_4, keeping the other values of $\mathbf{x}$ fixed at their optimal positions. The dissimilarities in this example have been discussed by Robinson (1951) and Hubert and Arabie (1986). It is easily seen that the (sub)gradient only changes whenever x_4 passes one of the other points. This step function indicates that $\rho(\mathbf{x})$ is a piecewise linear function, its pieces being linear within each rankorder of $\mathbf{x}$. For each rankorder, the STRESS is consequently quadratic in $\mathbf{x}$. It suggests that the unidimensional scaling problem can be solved by minimizing STRESS over all permutations, i.e., a combinatorial problem. A more profound discussion follows shortly.

Let ψ denote the rankorder of the vector $\mathbf{x}$, thus $x_{\psi(1)} \leq x_{\psi(2)} \leq \ldots \leq x_{\psi(i)} \leq \ldots \leq x_{\psi(n)}$ with corresponding permutation matrix $\mathbf{P}$, so that $\mathbf{Px}$ is the vector with the elements ordered nondecreasingly. We are going to show that at a local optimum of a function that is only dependent on the permutation ψ, the Guttman transform yields an $\mathbf{x}$ that has the rankorder given by ψ. Therefore, for this rankorder, STRESS has a local minimum. For the moment, let us assume that all weights $w_{ij} = 1$, which makes $\mathbf{V} = n\mathbf{I} - \mathbf{11}' = n\mathbf{J}$. Define $t_i = \sum_{j<i}\delta_{\psi(i)\psi(j)}$ and $u_i = \sum_{j>i}\delta_{\psi(i)\psi(j)}$, which is respectively the row sum up to the main diagonal and the row sum from the main diagonal of the matrix with values $\delta_{\psi(i)\psi(j)}$. Using this notation, (1.29) can be written as

$$\sigma^2(\mathbf{x}) = \eta_\delta^2 + \mathbf{x}'\mathbf{Vx} - 2\mathbf{x}'\mathbf{P}'(\mathbf{t}-\mathbf{u}). \tag{1.30}$$

For a *given* rankorder ψ, (1.30) is quadratic in $\mathbf{x}$ and has its minimum when $\mathbf{x}$ is equal to the Guttman transform $\mathbf{V}^{-}\mathbf{P}'(\mathbf{t} - \mathbf{u}) = n^{-1}\mathbf{JP}'(\mathbf{t} - \mathbf{u})$. The Guttman transform of the majorization approach only uses the rankorder information of the previous configuration, since $\mathbf{P}$, $\mathbf{t}$ and $\mathbf{u}$ only depend on the permutation of $\mathbf{x}$. Therefore, SMACOF will stop if the rankorder of $\mathbf{x}$ does not change, which usually happens in a few iterations. At this point STRESS has a local minimum. Function (1.30) can also be expressed as

$$\sigma^2(\mathbf{x}) = \eta_\delta^2 + n^{-1}\|n\mathbf{JPx} - (\mathbf{t} - \mathbf{u})\|^2 - n^{-1}\|\mathbf{t} - \mathbf{u}\|^2, \tag{1.31}$$

where the term $t^2(\psi) = n^{-1}\|\mathbf{t} - \mathbf{u}\|^2$ is a function of the permutation only. Thus if $t^2(\psi)$ is maximized, the second term of (1.31) vanishes for $\mathbf{x}$ equal to $n^{-1}\mathbf{JP}'(\mathbf{t} - \mathbf{u})$. Defays (1978) attacks the minimization of (1.31) by the maximization of $t^2(\psi)$. Suppose that we have found a permutation ψ that is locally optimal with respect to adjacent pairwise interchanges; i.e., any local change of ψ, interchanging $\psi(i)$ and $\psi(i+1)$, does not increases the value of $t^2(\psi)$. We say that $t^2(\psi)$ has a local maximum if permutation ψ satisfies this sufficient condition. Note that this is a stronger formulation for a local minimum than we used for STRESS, since STRESS has a local minimum whenever the Guttman transform cannot change the order of $\mathbf{x}$. We shall prove that at a local maximum of $t^2(\psi)$ the vector $\mathbf{t} - \mathbf{u}$ is ordered with increasing values. Clearly, this order is not disturbed by the Guttman transform, because $\mathbf{Px}$ has the same order. Thus if we can find a permutation ψ for which $n^{-1}\|\mathbf{t} - \mathbf{u}\|^2$ is locally optimal with respect to pairwise interchanges, then a local minimum of STRESS can be found in one step by taking the Guttman transform.

It remains to be proven that $t_i - u_i \leq t_{i+1} - u_{i+1}$ at a local maximum of $t^2(\psi)$. Suppose that t_i^* and u_i^* denote the values at such a local maximum and $t_i^{\sim}$ and $u_i^{\sim}$ denote the values of the same permutation except for one pairwise interchange of objects i and $i+1$. Since $n^{-1}\|\mathbf{t}^* - \mathbf{u}^*\|^2$ is a local maximum, it must be true that $\|\mathbf{t}^* - \mathbf{u}^*\|^2 \geq \|\mathbf{t}^{\sim} - \mathbf{u}^{\sim}\|^2$, or equivalently that

$$\left(t_i^* - u_i^*\right)^2 + \left(t_{i+1}^* - u_{i+1}^*\right)^2 \geq \left(t_i^{\sim} - u_i^{\sim}\right)^2 + \left(t_{i+1}^{\sim} - u_{i+1}^{\sim}\right)^2. \tag{1.32}$$

By definition we have

$$\begin{aligned} t_i^{\sim} &= t_{i+1}^* - \delta_{\psi(i)\psi(i+1)}, \\ t_{i+1}^{\sim} &= t_i^* + \delta_{\psi(i)\psi(i+1)}, \\ u_i^{\sim} &= u_{i+1}^* + \delta_{\psi(i)\psi(i+1)}, \\ u_{i+1}^{\sim} &= u_i^* - \delta_{\psi(i)\psi(i+1)}. \end{aligned} \tag{1.33}$$

Rewriting the right part of (1.32) gives

$$\begin{aligned} \left(t_i^* - u_i^*\right)^2 + \left(t_{i+1}^* - u_{i+1}^*\right)^2 \geq \left(t_{i+1}^* - u_{i+1}^* - 2\delta_{\psi(i)(i+1)}\right)^2 + \left(t_i^* - u_i^* + 2\delta_{\psi(i)(i+1)}\right)^2 = \\ \left(t_{i+1}^* - u_{i+1}^*\right)^2 + \left(t_i^* - u_i^*\right)^2 + 8\delta_{\psi(i)(i+1)}^2 + \\ 4\delta_{\psi(i)(i+1)}\left(t_i^* - u_i^*\right) - 4\delta_{\psi(i)(i+1)}\left(t_{i+1}^* - u_{i+1}^*\right). \end{aligned} \tag{1.34}$$

or, equivalently

$$4\delta_{\psi(i)\psi(i+1)}\left(2\delta_{\psi(i)\psi(i+1)} + \left(t_i^* - u_i^*\right) - \left(t_{i+1}^* - u_{i+1}^*\right)\right) \leq 0. \tag{1.35}$$

From (1.35) and the observation that $\delta_{\psi(i)\psi(i+1)} \geq 0$, it is easily verified that $t_i^* - u_i^* \leq t_{i+1}^* - u_{i+1}^*$, which holds for every i between 1 and n - 1. This result is due to Defays (1978), also, see Hubert and Arabie (1986) who call every permuted matrix with elements $\delta_{\psi(i)\psi(j)}$ monotonic, if $\mathbf{t} - \mathbf{u}$ is monotone nondecreasing. Note that the proof given above is only valid if all weights are equal to one.

If the weights w_{ij} are different from unity, $\mathbf{V}$ does not equal $n\mathbf{I} - \mathbf{11}'$, but much of what is said above still holds in a different metric. The STRESS function can be appropriately written as

$$\sigma^2(\mathbf{x}) = \eta_\delta^2 + \left\|\mathbf{x} - \mathbf{V}^-\mathbf{P}'(\mathbf{t} - \mathbf{u})\right\|_{\mathbf{V}}^2 - \left\|\mathbf{t} - \mathbf{u}\right\|_{\mathbf{PV}^-\mathbf{P}'}^2, \tag{1.36}$$

where norm $\|\mathbf{x}\|_{\mathbf{V}}^2$ equals $\mathbf{x}'\mathbf{V}\mathbf{x}$, $t_i = \sum_{j<i} a_{\psi(i)\psi(j)}$ and $u_i = \sum_{j>i} a_{\psi(i)\psi(j)}$ with $a_{ij} = w_{ij}\delta_{ij}$. Here too, it suffices to maximize $\|\mathbf{t} - \mathbf{u}\|_{\mathbf{PV}^-\mathbf{P}'}^2$ over all permutations and set $\mathbf{x} = \mathbf{V}^-\mathbf{P}'(\mathbf{t} - \mathbf{u})$. For this we have to establish that the permutation that maximizes $\|\mathbf{t} - \mathbf{u}\|_{\mathbf{PV}^-\mathbf{P}'}^2$ gives a vector $\mathbf{PV}^-\mathbf{P}'(\mathbf{t} - \mathbf{u})$ that is monotonic increasing, so that taking the Guttman transform does not change the rankorder of $\mathbf{x}$, which implies a local minimum for STRESS. This brings us to the following theorem.

Theorem 1.2
Minimizing STRESS with $p = 1$, $w_{ij} \geq 0$ and $\mathbf{W}$ irreducible, can be solved by maximizing $t^2(\psi) = \|\mathbf{t} - \mathbf{u}\|_{\mathbf{PV}^-\mathbf{P}'}^2$ over all permutations ψ. If $t^2(\psi)$ is maximal with respect to all permutations that are adjacent pairwise interchanges of ψ^*, then STRESS has a local minimum.

Proof
Suppose that we have found a permutation ψ^* for which $\|\mathbf{t} - \mathbf{u}\|_{\mathbf{PV}^-\mathbf{P}'}^2$ is optimal with respect to all adjacent pairwise interchanges, i.e., $\|\mathbf{t}^* - \mathbf{u}^*\|_{\mathbf{PV}^-\mathbf{P}'}^2 \geq \|\mathbf{t}^\sim - \mathbf{u}^\sim\|_{\mathbf{QV}^-\mathbf{Q}'}^2$, where $\mathbf{Q}$ is the permutation matrix based on $\psi^\sim$. Let $\mathbf{A}$ be the matrix $(\mathbf{e}_i - \mathbf{e}_{i+1})(\mathbf{e}_i - \mathbf{e}_{i+1})'$, with $\mathbf{e}_i$ the ith column of $\mathbf{I}$, so that the permutation matrix $\mathbf{I} - \mathbf{A}$ can be used to interchange the values of elements i and i+1, i.e., $\mathbf{Q} = (\mathbf{I} - \mathbf{A})\mathbf{P}$. This makes

$$\|\mathbf{t}^\sim - \mathbf{u}^\sim\|_{\mathbf{QV}^-\mathbf{Q}'}^2 = \|\mathbf{t}^\sim - \mathbf{u}^\sim\|_{(\mathbf{I}-\mathbf{A})\mathbf{PV}^-\mathbf{P}'(\mathbf{I}-\mathbf{A})}^2 = \|(\mathbf{I} - \mathbf{A})(\mathbf{t}^\sim - \mathbf{u}^\sim)\|_{\mathbf{PV}^-\mathbf{P}'}^2. \tag{1.37}$$

We also know that $\mathbf{t}^\sim - \mathbf{u}^\sim = (\mathbf{I} - \mathbf{A})(\mathbf{t}^* - \mathbf{u}^*) - 2a_{\psi(i)\psi(j)}(\mathbf{e}_i - \mathbf{e}_{i+1})$, so that $(\mathbf{I} - \mathbf{A})(\mathbf{t}^\sim - \mathbf{u}^\sim) = (\mathbf{t}^* - \mathbf{u}^*) - 2a_{\psi(i)\psi(j)}(\mathbf{I} - \mathbf{A})(\mathbf{e}_i - \mathbf{e}_{i+1})$, which allows us to rewrite (1.37) as

$$\begin{aligned}\|\mathbf{t}^\sim - \mathbf{u}^\sim\|_{\mathbf{QV}^-\mathbf{Q}'}^2 = \quad & \|\mathbf{t}^* - \mathbf{u}^*\|_{\mathbf{PV}^-\mathbf{P}'}^2 + 4a_{\psi(i)\psi(j)}^2 \|(\mathbf{I} - \mathbf{A})(\mathbf{e}_i - \mathbf{e}_{i+1})\|_{\mathbf{PV}^-\mathbf{P}'}^2 \\ & - 4a_{\psi(i)\psi(j)}(\mathbf{t}^* - \mathbf{u}^*)'\mathbf{PV}^-\mathbf{P}'(\mathbf{I} - \mathbf{A})(\mathbf{e}_i - \mathbf{e}_{i+1}).\end{aligned} \tag{1.38}$$

The second and third term in (1.38) can be simplified using $(\mathbf{I} - \mathbf{A})(\mathbf{e}_i - \mathbf{e}_{i+1}) = (\mathbf{e}_i - \mathbf{e}_{i+1}) - (\mathbf{e}_i - \mathbf{e}_{i+1})(\mathbf{e}_i - \mathbf{e}_{i+1})'(\mathbf{e}_i - \mathbf{e}_{i+1}) = (\mathbf{e}_i - \mathbf{e}_{i+1}) - 2(\mathbf{e}_i - \mathbf{e}_{i+1}) = -(\mathbf{e}_i - \mathbf{e}_{i+1})$, so that the second term becomes

$$4a^2_{\psi(i)\psi(j)} \|(\mathbf{I} - \mathbf{A})(\mathbf{e}_i - \mathbf{e}_{i+1})\|^2_{\mathbf{PV}^-\mathbf{P}'} = 4a^2_{\psi(i)\psi(j)} \|\mathbf{e}_i - \mathbf{e}_{i+1}\|^2_{\mathbf{PV}^-\mathbf{P}'}, \tag{1.39}$$

and the third term becomes

$$\begin{aligned} &- 4a_{\psi(i)\psi(j)} (\mathbf{t}^* - \mathbf{u}^*)'\mathbf{PV}^-\mathbf{P}'(\mathbf{I} - \mathbf{A}) (\mathbf{e}_i - \mathbf{e}_{i+1}) = \\ &\quad 4a_{\psi(i)\psi(j)} (\mathbf{t}^* - \mathbf{u}^*)'\mathbf{PV}^-\mathbf{P}'(\mathbf{e}_i - \mathbf{e}_{i+1}). \end{aligned} \tag{1.40}$$

Summarizing these results gives

$$\begin{aligned} \|\mathbf{t}^* - \mathbf{u}^*\|^2_{\mathbf{PV}^-\mathbf{P}'} \geq \quad & \|\mathbf{t}^\sim - \mathbf{u}^\sim\|^2_{\mathbf{QV}^-\mathbf{Q}'} = \\ & \|\mathbf{t}^* - \mathbf{u}^*\|^2_{\mathbf{PV}^-\mathbf{P}'} + 4a^2_{\psi(i)\psi(j)} \|\mathbf{e}_i - \mathbf{e}_{i+1}\|^2_{\mathbf{PV}^-\mathbf{P}'} \\ & + 4a_{\psi(i)\psi(j)} (\mathbf{t}^* - \mathbf{u}^*)'\mathbf{PV}^-\mathbf{P}'(\mathbf{e}_i - \mathbf{e}_{i+1}), \end{aligned} \tag{1.41}$$

or equivalently

$$4a^2_{\psi(i)\psi(j)} \|\mathbf{e}_i - \mathbf{e}_{i+1}\|^2_{\mathbf{PV}^-\mathbf{P}'} + 4a_{\psi(i)\psi(j)} (\mathbf{t}^* - \mathbf{u}^*)'\mathbf{PV}^-\mathbf{P}'(\mathbf{e}_i - \mathbf{e}_{i+1}) \leq 0. \tag{1.42}$$

Since $a_{\psi(i)\psi(j)}$ and $\|\mathbf{e}_i - \mathbf{e}_{i+1}\|^2_{\mathbf{PV}^-\mathbf{P}'}$ are nonnegative, we must have $(\mathbf{t}^* - \mathbf{u}^*)'\mathbf{PV}^-\mathbf{P}'(\mathbf{e}_i - \mathbf{e}_{i+1}) \leq 0$ for (1.42) to hold. This implies that the elements of $\mathbf{PV}^-\mathbf{P}'(\mathbf{t}^* - \mathbf{u}^*)$ are monotonically increasing. If $\mathbf{x}^*$ equals the Guttman transform $\mathbf{V}^-\mathbf{P}'(\mathbf{t}^* - \mathbf{u}^*)$, it follows that $\mathbf{Px}^*$ is also monotonically increasing, so that the rankorder does not change and hence STRESS has a local minimum. Q.E.D.

Heiser (1981) showed that metric unfolding can be formulated as a multidimensional scaling problem with a special weight matrix $\mathbf{V}$ (also, see section 5.2). In turn, the result above shows that unidimensional unfolding is a combinatorial problem, which was also recognized by Heiser (1981) and Poole (1984, 1990) in a different way. We return to this subject in chapter 5.

We have shown that unidimensional scaling amounts to maximizing a measure over all possible permutations, i.e., a combinatorial problem. We also demonstrated that the Guttman transform uses the rankorder information only and is finished whenever the rankorder does not change, usually within a few iterations. Therefore, whenever $p = 1$ it seems fruitful to switch to a combinatorial optimization strategy.

1.6 Non-metric scaling

In the previous part, the dissimilarities were kept fixed. However, this is not strictly necessary; we may also use only the *order* information of the dissimilarities. The main contribution of Shepard (1962) and Kruskal (1964a,b) was that they realized this very fact and proposed appropriate algorithms that allow for non-metric transformations of the

dissimilarities. Using transformations that are monotonic with the dissimilarities relaxes the STRESS model. At the cost of loosing the direct relation between dissimilarities and distances, non-metric scaling yields lower STRESS then metric MDS, so that usually an acceptable solution is obtained in less dimensions. Apart from the monotone regression transformation, many other transformations can be used. We return to this in a moment.

Non-metric scaling can be viewed in two ways. First, Kruskal (1977) views it as a function of the distances: for every matrix of distances and given rankorder of δ_{ij}, there exists a matrix of optimally transformed dissimilarities, also called pseudo-distances . The second way of viewing non-metric scaling is described by De Leeuw and Heiser (1977) and De Leeuw (1988) who regard it as an alternating least squares algorithm: in one step the pseudo-distances are fixed and a better fitting configuration is sought, in the other step the configuration is fixed and the pseudo-distances are optimally transformed. Sometimes we have to impose additional constraints on the pseudo-distances to avoid degenerate solutions. For example, when using monotone regression it is necessary to normalize the pseudo-distances with sum of squares 1 (or any other constant) to avoid a solution where all pseudo-distances and all distances are zero.

Generally, any class of transformations can be used that belongs to a model that makes sense. Suppose that instead of fixed dissimilarities only intervals for each dissimilarity is given. Then we could define the pseudo-distance being as close as possible to the corresponding distance, provided it remains in the supplied interval. Other transformations like B-spline transformation, linear regression, multiple regression, or polynomial regression could also be used. Depending on the transformation, negative pseudo-distances may occur, for example when using linear regression with an intercept, which requires an adaptation of the SMACOF algorithm (see Heiser, 1991) to retain convergence.

1.7 Special MDS models

Various models known from the literature can be implemented by imposing restrictions on $\mathbf{X}$. An extensive discussion for metric projection restrictions in the SMACOF algorithm has been given by De Leeuw and Heiser (1980). Often, but not always, analytic solutions for metric projections exist. Even if no analytic solution exists, we show below that the sequence of STRESS values remains convergent.

A restriction can apply elementwise, rowwise, columnwise, on the configuration as a whole, or combinations of this. An example of elementwise restrictions is the use of equality constraints among various elements of $\mathbf{X}$. Rowwise restrictions may be used to constrain all points to be on the surface of a p dimensional sphere. Column restrictions may be used to generate a configuration matrix with $\mathbf{X}'\mathbf{X} = \mathbf{I}$. Finally, the size of the configuration may be restricted by requiring that $\text{tr}\mathbf{X}'\mathbf{X} = c$ as done by Mathar (1989). An important class of restrictions is formed by restrictions for which an analytic solution

exists of the metric projection P_Ω, that finds for any $\mathbf{Y}$ an $\mathbf{X} \in \Omega$ as close as possible to $\mathbf{Y}$, i.e.,

$$\left\|\mathbf{X}^+ - \overline{\mathbf{X}}\right\|_V^2 = \min_{\mathbf{Y}\in\Omega} \left\|\mathbf{Y} - \overline{\mathbf{X}}\right\|_V^2, \tag{1.43}$$

where $\overline{\mathbf{X}}$ is the Guttman transform $\mathbf{V}^-\mathbf{B}(\mathbf{Y})\mathbf{Y}$. We shall now discuss some known restrictions. Many special models and applications of restrictions could be given, but we restrict ourselves to some powerful examples. Important models for three way analysis are the INDSCAL model and the IDIOSCAL model of Carroll and Chang (1970). In the context of MDS we have a dissimilarity matrix for each layer k, whose configuration $\mathbf{X}_k$ should be related to a common configuration $\mathbf{X}$ by $\mathbf{X}_k = \mathbf{X}\mathbf{S}_k$. For diagonal $\mathbf{S}_k$ the model is named INDSCAL, and if $\mathbf{S}_k$ is positive semi-definite we deal with the IDIOSCAL model. Another application making wide use of ordered cone restrictions is the non-linear extension of multivariate analysis in the STRESS framework of Meulman (1986, 1992). There, order restrictions are imposed on columns. A third possible application of restrictions is to require that $\mathbf{X}$ is a linear combination of some previous estimates and the Guttman transform. In this way De Leeuw and Heiser (1980) created a conjugate gradient method. Generally, if $\mathbf{X}$ has to satisfy multiple restrictions Ω_j, where each Ω_j is convex, we may use the cyclic projection algorithm of Dykstra (1983). The cyclic projection algorithm has been used extensively for a variety of restrictions by Van der Lans (1992).

From the majorizing function of $\sigma^2(\mathbf{X})$, the Guttman transform $\overline{\mathbf{X}}$, and any restricted configurations $\mathbf{X}^+$,$\mathbf{Y}$ for which

$$\left\|\mathbf{X}^+ - \overline{\mathbf{X}}\right\|_V^2 \leq \left\|\mathbf{Y} - \overline{\mathbf{X}}\right\|_V^2 \tag{1.44}$$

holds, it can be shown that

$$\sigma(\mathbf{X}^+) \leq \eta_\delta^2 + \left\|\mathbf{X}^+ - \overline{\mathbf{X}}\right\|_V^2 - \left\|\overline{\mathbf{X}}\right\|_V^2 \leq \sigma(\mathbf{Y}). \tag{1.45}$$

This can be seen by adding η_δ^2 and $\left\|\overline{\mathbf{X}}\right\|_V^2$ to both sides of (1.44), which gives

$$\sigma(\mathbf{X}^+) \leq \eta_\delta^2 + \left\|\mathbf{X}^+ - \overline{\mathbf{X}}\right\|_V^2 - \left\|\overline{\mathbf{X}}\right\|_V^2 \leq \eta_\delta^2 + \left\|\mathbf{Y} - \overline{\mathbf{X}}\right\|_V^2 - \left\|\overline{\mathbf{X}}\right\|_V^2 = \sigma(\mathbf{Y}). \tag{1.46}$$

As a consequence we get a convergent series of STRESS values for such restricted configurations, even though (1.43) need not be satisfied. This opens the possibility of using a wide class of restrictions. For some restrictions, like cluster restrictions, it is possible to find an $\mathbf{X}^+$ that meets the weaker condition (1.44), but it is very hard to have it satisfy (1.43).

1.8 Outline of this monograph

From the previous sections it should be clear that much is known about algorithms for STRESS and applications of STRESS. However, some issues remain open. The elegant method of majorization only guarantees lower STRESS values, and stops usually at a local minimum, which need not be a global minimum. This problem is especially severe for the unidimensional case. Therefore, we study methods that aim at finding the global minimum. This is done in chapter 2. Then we concentrate on one special method, called the tunneling method, in chapter 3 and apply it to the STRESS function. Minimization by the tunneling method is implemented with iterative majorization. In chapter 4 we try to find indications when local minima may be found, and get an idea of the performance of some global optimization methods.

A useful aspect of the STRESS function is the inclusion of weights. It allows for missing patterns with multidimensional scaling. Chapter 5 discusses several structured missing data patterns. In particular, we discuss acceleration of inverting a matrix associated with the STRESS function. Also, we indicate how missing data patterns can be used for (external) unfolding and semi-complete scaling. Inspecting the multidimensional scaling solution can be difficult with a large number of objects. Therefore, restrictions can be helpful to reduce the amount of output. In chapter 6, we discuss (fuzzy) clustering implementations in MDS. Finally, in chapter 7 we extend the majorization method for STRESS to include Minkowski distances.

CHAPTER 2

GLOBAL OPTIMIZATION AND STRESS

One of the advantages of the majorization algorithm in metric multidimensional scaling is that a monotone decreasing series of STRESS values is obtained. The majorization method also has a weakness; the solution is usually a locally optimal solution, but not necessarily the best overall solution. Thus, other configurations may exist that have a better fit. Finding the best overall solution corresponds in mathematical terms to finding a *global minimum* of the STRESS function, which is an important topic in this monograph. Several *local* search procedures for STRESS are well known, like the gradient method proposed by Kruskal (1964a,b) and the SMACOF algorithm of De Leeuw (1977) and De Leeuw and Heiser (1980). In fact, any standard nonlinear programming technique known from the field of numerical optimization could be used for a local search. Such local search techniques give at best a local minimum, but not a global one. The problem of finding global minima is the main concern of the field of global optimization. Unfortunately, the knowledge of existence of a lower minimum is rarely available.

The mathematical formulation of a global minimum $\mathbf{X}^*$ of the real-valued function $\sigma(\mathbf{X})$ is that

$$\sigma(\mathbf{X}^*) - \sigma(\mathbf{X}) \leq 0 \tag{2.1}$$

for *all* feasible $\mathbf{X}$. This does not imply that $\mathbf{X}^*$ is unique; there may well be other $\mathbf{X}$ with $\sigma(\mathbf{X}^*) = \sigma(\mathbf{X})$. In contrast, a local minimum is defined to be the point where all other points *within a small neighbourhood* have higher function values. Thus, a global minimum is the lowest local minimum. Apart from the necessary conditions for a local minimum like a vanishing gradient and a matrix of second derivatives that is positive semi-definite, no such mathematical condition exists for the global minimum (except equivalent formulations of (2.1)). Unless the function exhibits special analytical properties it will generally be hard to find the global optimum. Horst and Tuy (1990) warn that "... global solution methods must be significantly different from non-linear programming techniques, and they can be expected to be - and are - much more expensive computationally."

An aspect which makes global optimization difficult is the so called *curse of dimensionality*. As the number of parameters to be estimated rises, so does the space to be searched. In fact, the amount of space to be searched grows exponentially with the number of parameters. Even for a small number of parameters an exhaustive search is

virtually impossible, unless a very large amount of the space can be discarded from the start. The number of parameters for STRESS, which is the total number of coordinates of the configuration, is very high. Therefore, it may be expected that finding the global minimum of the STRESS function is (computationally) very hard.

In the next section we give a brief overview of the literature on global optimization, and, where possible, in the context of minimizing the STRESS function. Recently there has been an increasing interest in global optimization. Some recent publications reviewing this area are Horst and Tuy (1990), Rinnooy Kan and Timmer (1987a,b), and Törn and Žilinskas (1989). Moreover, in 1991 the Journal of Global Optimization was founded. In chapter 1 we saw that there are three special cases of MDS: full-dimensional scaling, that has no local minimum problem, unidimensional scaling, that has an especially severe local minimum problem, and MDS with $1 < p < n - 1$, for which the local minimum problem can also be present. For unidimensional scaling we apply some algorithms especially suited for combinatorial problems. Next, we look at methods better suited for MDS with $p > 1$, such as stochastic methods, like the well-known multistart method and multi-level-single-linkage, which can be seen as an improvement of multistart. An important contribution of this monograph is the tunneling method for global optimization, discussed extensively in chapter 3.

We are mainly concerned with algorithms that aim at finding a global minimum within a reasonable amount of time. Some of the global optimization methods discussed below consist of a local search phase and a global search phase. We use the SMACOF algorithm during the local search.

2.1 A classification of the field of global optimization

A short summary and classification of global optimization techniques will be given here and some of them discussed in more detail in subsequent sections. We do not intend to cover the field exhaustively, but we wish to create a framework in which the global optimization techniques discussed can be placed. Several classifications of this field exist, but we follow the classification given by Törn and Žilinskas (1989). They separate the techniques into those with a guaranteed accuracy in reaching the global minimum, and those without (see Figure 2.1). To the former group belong *covering* methods, that exclude subregions that do not contain the global minimum. The techniques without guaranteed accuracy are separated in *direct* methods (random search, clustering methods and generalized descent methods) that use only local information and *indirect* methods (methods approximating the level sets, or the objective function) that build a model of the level sets or the objective function.

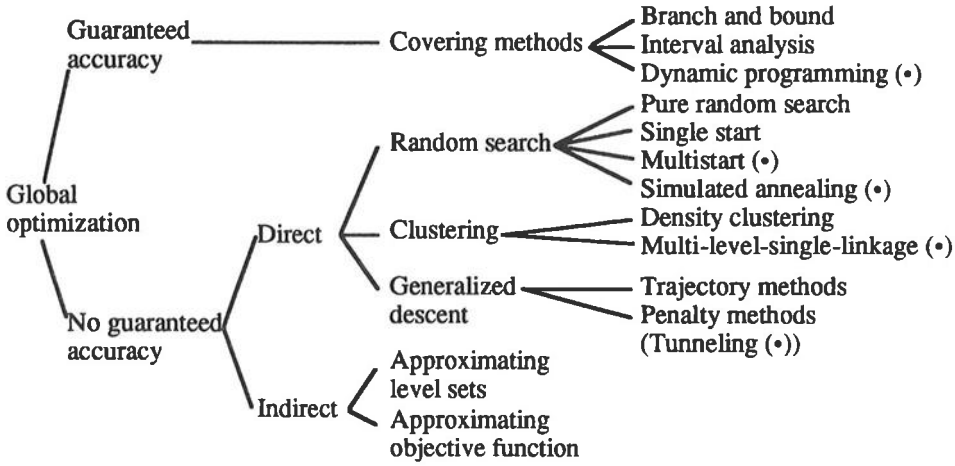


Figure 2.1 *Classification of global optimization methods. Methods indicated with (•) are discussed in more detail.*

A simple example of a *covering method* is complete enumeration in combinatorial problems. Branch and bound techniques also fall in this class. Here the total parameter space is split in subsets for which lower and upper bounds of the function are known. The subsets that are known not to contain the global minimum are excluded. The remaining subsets are split again, so that we end up with the global minimum within some pre-set accuracy. Recent publications of this type of methods is Horst and Tuy (1990), who only use convex analysis and do not need the usual requirement of existing first and second derivatives of the objective function. Klaassen (1989) has applied a variant of this method, interval analysis (see Hansen, 1980), to the STRESS function. Unfortunately, his method was only successful for very small problems. Another type of covering method was used by Hubert and Arabie (1986), who used dynamic programming successfully for unidimensional scaling (see section 2.2.1). We return to covering methods in section 2.3. These methods can be regarded as deterministic, since they have guaranteed success. Note that this does not imply that they succeed in finite time.

Under the direct methods we find the *random search* methods like pure random search, singlestart and multistart (see section 2.4). In all three methods random configurations are generated and STRESS values determined. In single start a local search is started from a random configuration with the lowest STRESS, whereas multistart starts a local search from every random configuration. No local search is used in the pure random search. All three methods consider the lowest STRESS value to be the global minimum. For an overview of pure random search see Zhigljavsky (1991). These strategies (especially multistart) are widely used to investigate whether a function suffers from local minima or not. Since the early 1980's a different random search method

became popular; the method of simulated annealing. During the simulated annealing process a different random perturbation is generated. This perturbation is always accepted if it yields a decrease in function value. However, an increase in function value is also accepted with a certain (decreasing) probability. Machmouchi (1992) applied simulated annealing to non-metric MDS, but little is known about its efficiency in determining a global minimum. We discuss simulated annealing for unidimensional scaling in more detail in section 2.2.3.

The third type is formed by *clustering* methods. These methods refine the multistart method in that they intend to start a local search only once for configurations leading to the same local minimum (Rinnooy Kan and Timmer, 1987a,b). This type of algorithm is often regarded as relatively efficient (Zhigljavsky, 1991; Törn and Žilinskas; 1989). Two examples of this method are Törn's density clustering (1976) and the multi-level-single-linkage algorithm of Timmer (1984). We adapt Timmer's clustering method to MDS in section 2.5.

The fourth type is formed by *generalized descent* methods, where the function is modified to guarantee a lower minimum. Trajectory methods try to model the stationary points (local minima, local maxima and saddle points) by a curve in an extra parametrization. Following the trajectory of the curve would lead to all local minima, including the global one. These methods are commonly not regarded as being promising (Timmer, 1984; Törn and Žilinskas; 1989). Penalty methods modify the function itself to prevent returning to local minima found in previous iterations. One such penalty method is the tunneling algorithm, that alternates a local search with a tunneling step. In the latter step we try to find a solution different from the previous local minimum, that has the same STRESS value as the local minimum. An important and attractive feature of the tunneling algorithm is that successive local minima are always lower. We return to this method in more detail in chapter 3.

The first indirect models aim at *approximating the level sets*, that is the set of configurations with STRESS smaller or equal to some constant. The level set L_c is the set $\{\mathbf{X} \in \mathbb{R}^{n \times p} \mid \sigma(\mathbf{X}) \leq c\}$. The idea is to find a c and thus a solution $\mathbf{X}$ for which a volume measure of the set L_c equals zero. At such a configuration STRESS must have its global minimum. Törn and Žilinskas note that this approach does not seem to be very promising.

The last type is formed by methods *approximating the objective function*. Here a theory is developed based on a statistical model of the function. The unknown function values are treated as random variables in a Bayesian analysis. This class of methods can be used for functions that are expensive to evaluate, so that all information is used optimally. It has been found very efficient for oscillating functions of one parameter only (see Žilinskas, 1978) and for engineering problems with functions that are expensive to evaluate. For more details we refer to Mockus (1989).

Now that we have established a framework of global optimization methods, we apply some of these methods to the STRESS function. First, we discuss unidimensional

scaling with $p = 1$. The global optimization methods that follow thereafter are best suited for $1 < p < n - 1$, although they may also be applied to unidimensional scaling.

2.2 Unidimensional scaling

We showed in section 1.5 that minimizing the STRESS function for $p = 1$ amounts to a combinatorial problem. A nice implication of this observation is that finding a global maximum of $t^2(\psi)$ must also give a global minimum of STRESS. Thus we have to avoid local maxima of the function $t^2(\psi)$. Defays (1978) used a branch and bound algorithm to find the global minimum of STRESS. Another obvious strategy is to look at all permutations and select the optimal one. This strategy is prohibitive for large n, since we have to deal with $n!$ different permutations. Therefore, we discuss more efficient algorithms, like the dynamic programming approach of Hubert and Golledge (1981) and Hubert and Arabie (1986), which finds the global optimum for small sized n (say $n <$ 20). For larger n, this approach is too intensive, so we have to change to searches that are not necessarily guaranteed to arrive at the global optimum. Of these we discuss pairwise interchange strategies, simulated annealing, and tabu-search, which aims at finding increasingly better local optima.

2.2.1 Dynamic programming

In the previous chapter, it turned out that unidimensional scaling is a combinatorial problem. Hubert and Golledge (1981) and Hubert and Arabie (1986) used a special feature of $t^2(\psi)$ to reduce the order of the computations needed to obtain a global maximum from $n!$ to 2^n. Let R be a subset of m–1 of the n objects and R' be the subset of m objects obtained by adding object k that was not in R. If object k is at position m in the final permutation, of which the first m–1 are given by ψ_{m-1}, then its contribution to the final value of $\|\mathbf{t} - \mathbf{u}\|^2$ is given by $(t_m - u_m)^2 = (\sum_{j \in R}\delta_{jk} - \sum_{j \notin R}\delta_{jk})^2$. Suppose that $f(R)$ contains the maximum contributions to $t^2(.)$ so far. Then the maximum contribution including object k in position m is

$$f(R') = \max f(R) + \left(\sum_{j \in R}\delta_{k,\psi(j)} - \sum_{j \notin R}\delta_{k,\psi(j)}\right)^2. \tag{2.2}$$

over all $m = \binom{m}{m-1}$ subsets R and objects k that form together subset R'. Thus to assess the contribution of object k in position m to the optimal $t^2(.)$ we need to know neither the particular order of ψ_m nor which objects are positioned after m. We only need to know which objects are positioned before m, not their particular order. The empty set with no objects present ($m = 0$) has $f(\varnothing) = 0$.

The recursive relation in (2.2) is exploited in the dynamic programming approach. We start with the empty set and consider $\binom{n}{1}$ subsets of size 1 and store the value of (2.2) for each of them. Then we regard the $\binom{n}{2}$ subsets of size 2, store their value of (2.2) and also store the object that was added last to obtain $f(R')$. This is repeated for each larger subset until the final set of all objects is considered. Then we use the two arrays for finding the permutation that led to the maximum solution, starting from the last added object to the first one.

Non-identical weights

The same approach as in dynamic programming can be extended to unidimensional scaling with general nonnegative weights. However, we cannot guarantee a global minimum anymore. We have seen that for non-identical weights, $\|\mathbf{t}-\mathbf{u}\|^2_{\mathbf{PV}^-\mathbf{P}'}$ has to be optimized. For this function it is still possible to judge the contribution of object k in position m using only the objects in the positions smaller than m and object k. However, in contrast to the unit weights case, we do have to know the ordering of the objects in positions smaller than m. Hence some of the efficiency of the dynamic programming approach is lost, although we still only have to consider 2^n subsets.

Let $\mathbf{PV}^-\mathbf{P}'$ be decomposed by a Cholesky factorization in $\mathbf{RR}'$, with $\mathbf{R}$ an upper triangular matrix. This can be done by finding the ordinary Cholesky factorization $\mathbf{R}'\mathbf{R}$ of matrix $\mathbf{PV}^-\mathbf{P}'$ transposed over the lower-left upper-right diagonal. There should be no problem in finding such decomposition since V is positive semi-definite, hence so is $\mathbf{PV}^-\mathbf{P}'$. Then we can show that the contribution of object k in position m to $\|\mathbf{t}-\mathbf{u}\|^2_{\mathbf{PV}^-\mathbf{P}'}$ depends only on the values of $t_i - u_i$ for $i \le m$. Note that $\|\mathbf{t}-\mathbf{u}\|^2_{\mathbf{PV}^-\mathbf{P}'} = ((\mathbf{t}-\mathbf{u})\mathbf{R})'((\mathbf{t}-\mathbf{u})\mathbf{R})$, where $(\mathbf{t}-\mathbf{u})\mathbf{R}$ is a vector whose element i is only dependent on elements $j \le i$. The function to be maximized is

$$t^2(\psi) = \|\mathbf{t}-\mathbf{u}\|^2_{\mathbf{PV}^-\mathbf{P}'} = \sum_i \left(\sum_{j \le i} r_{ji}(t_j - u_j) \right)^2 . \tag{2.3}$$

The maximum contribution to $t^2(.)$ of having object k in position m is

$$f(R') = \max f(R) + \left(\sum_{j \le m} r_{jm}(t_j - u_j) \right)^2 . \tag{2.4}$$

over all $m = \binom{m}{m-1}$ subsets R and objects k that form together subset R', knowing the values $t_i - u_i$ for $i < m$. The value of $t_m - u_m = \sum_{j \in R} a_{k,\psi(j)} - \sum_{j \notin R} a_{k,\psi(j)}$.

We can work our way recursively until we have subset R' containing all objects. For each different subset we have to store three items, the value $f(R)$, the object that was added last to obtain R, and the value of $t_i - u_i$ and we have to compute the Cholesky factorization to obtain $\mathbf{R}$. Note that we only need the part of $\mathbf{R}$ determined by the elements $\psi(j)$ with $j \le i$. We start with the empty subset and work our way to the subset of all objects using

(2.4). However, in this way we cannot guarantee that the global maximum of $\|\mathbf{t}-\mathbf{u}\|^2_{\mathbf{PV}^-\mathbf{P}'}$ is found in the last subset that contains all objects, due to violation of the principle of optimality in dynamic programming. This principle states that $f(R)$ must be optimal regardless of the object added to form R'. Thus the final permutation does not need to give the order of coordinates in the global minimum configuration of STRESS, although we expect it to be close to a global minimum. Compared with the unit weight case, the current procedure does increase the number of operations because the Cholesky decomposition of $\mathbf{PV}^-\mathbf{P}'$ has to be computed for each of the 2^n subsets.

2.2.2 *Pairwise interchange strategies*

For large n the dynamic programming approach of Hubert and Golledge (1981) becomes impractical. Therefore, we discuss in the following sections heuristic methods that find a local maximum relatively fast, some with mechanisms for avoiding local minima. Here we discuss pairwise interchange strategies, where some interchanges of two objects are compared and the one is chosen that increases $t^2(\psi)$ maximally.

A pairwise interchange strategy tries to find a permutation that is maximal with respect to some pairwise interchanges. We shall call this heuristic the *l*ocally *o*ptimal *p*airwise *i*nterchange (LOPI) strategy (cf. De Soete, Hubert, and Arabie, 1988). Let ψ_{ij} be the permutation resulting from ψ with pair i, j interchanged, i.e., $\psi_{ij}(k) = \psi(k)$ for all $k \neq i,j$ and $\psi_{ij}(i) = \psi(j)$, $\psi_{ij}(j) = \psi(i)$. Define the neighbourhood N of ψ to be those permutations that can be formed with one pairwise interchange from ψ. The local search of this strategy amounts to finding iteratively the maximum of $t^2(\psi_{ij})$ over N until $t^2(\psi_{ij})$ is no larger than $t^2(\psi)$ for all permutations in the neighbourhood N, so that ψ defines a local maximum. The attraction of pairwise interchange strategy lies in its simplicity and its computational efficiency in determining the best pairwise interchange. The efficiency is dependent on the definition of the neighbourhood N, of which we discuss four.

In its simplest form N consists of all pairwise interchanges of adjacent objects, thus considering $\psi_{i,i+1}$ for $i = 1, n-1$. We shall call this the LOPI1 strategy. One iteration of the LOPI1 strategy can be summarized by

$$\psi^{(k+1)} \leftarrow \arg\max_{i=1}^{n-1} t^2(\psi_{i,i+1}^{(k)}),$$

where the superscript (k) denotes the iteration number. To decide which pairwise interchange has to be carried out, we need to assess the $n-1$ values of $t^2(\psi_{i,i+1})$. It is not necessary to compute the $t^2(\psi_{i,i+1})$ from scratch, but instead it can be computed from $t^2(\psi)$ and using (1.33) for weights equal to one, or from $t^2(\psi)$ and using parts from (1.37) for non-identical weights.

A second strategy is advocated by Poole (1984, 1990) for unfolding. He considers each object separately and finds the optimal position for this object keeping the order of

the other objects fixed. Let us call this strategy LOPI2. One iteration of LOPI2 can be formulated as

$\psi'' \leftarrow \psi^{(k)}$
loop over objects $l = 1$ to n
    determine position i of object l
    $\psi'(1) \leftarrow \psi''(i)$, $\psi'(j+1) \leftarrow \psi''(j)$ for $j < i$ and $\psi'(j) \leftarrow \psi''(j)$ for $j > i$
    $\psi'' \leftarrow \psi'$
    loop over $j = 1$ to $n - 1$
        $\psi'' \leftarrow$ argmax ($t^2(\psi'')$, $t^2(\psi'_{j,j+1})$)
        $\psi' \leftarrow \psi'_{j,j+1}$
    end loop over j
end loop over l
$\psi^{(k+1)} \leftarrow \psi''$.

Note that this is a stronger formulation of a local maximum than LOPI1, where we stop whenever there is no adjacent pairwise interchange that gives a higher value of $t^2(\psi)$. At such a permutation it may still be possible that the pairwise interchange $\psi_{i,i+2}$ would increase the value of $t^2(\psi)$. LOPI2 excludes this case too. Thus the neighbourhood of LOPI2 is larger than that of LOPI1 and includes the neighbourhood of LOPI1. Clearly, the permutation that satisfies the stop condition of LOPI2, also satisfies the stopping condition of LOPI1. Poole (1990) conjectures that the number of permutations that satisfies the stopping criterion of LOPI2 decreases with n. If true, then LOPI2 would be a very successful strategy for large n. Just as for LOPI1 we can compute $t^2(\psi'_{j,j+1})$ quite efficiently in the (un)weighted case from $t^2(\psi')$.

Heiser (1989) uses pairwise interchanges in the context of city-block scaling. He arrives at local maxima with the same properties as LOPI1, but via a different path. Heiser regards the pairwise interchanges of all $n(n-1)/2$ pairs. We call Heiser's strategy LOPI3. It can be summarized as

$\psi' \leftarrow \psi^{(k)}$
loop over $i = 1$ to n
    loop over $j = 1$ to n
        $\psi' \leftarrow$ argmax ($t^2(\psi')$, $t^2(\psi'_{ij})$)
    end loop over j
end loop over i
$\psi^{(k+1)} \leftarrow \psi'$.

Indeed, if no change occurs it must be true that object i has optimal position keeping the order of the other objects fixed, so that a maximum of LOPI3 must also be a maximum of LOPI1, but not necessarily of LOPI2. LOPI3 is computationally more intensive than LOPI1, since more pairs have to be considered. An even more intensive variant of LOPI3 was used by De Soete et al. (1988). Their variant, which we call LOPI4, amounts to

$$\psi^{(k+1)} \leftarrow \operatorname{argmax}_{i<j} \left(t^2(\psi^{(k)}), t^2(\psi_{ij}^{(k)}) \right).$$

Again a permutation at the stopping condition of LOPI4 also satisfies the stopping condition of LOPI3.

In section 1.5 we showed that for ψ^* optimal with respect to adjacent pairwise interchanges, the vector $\mathbf{t} - \mathbf{u}$ in the identical weights case or the vector $\mathbf{P}\mathbf{V}^{-}\mathbf{P}'(\mathbf{t} - \mathbf{u})$ in the non-identical weights case are monotone increasing. Hence, STRESS has a local minimum for this permutation, because taking the Guttman transform does not change the order. If ψ^* is the permutation at the stop condition of LOPI2, LOPI3, or LOPI4, then it also satisfies the stop condition of LOPI1. Therefore, the LOPI2, LOPI3 and LOPI4 strategies also have a local minimum of STRESS for permutation ψ^*. Since LOPI2, LOPI3, and LOPI4 consider a larger neighbourhood for finding the local maximum of $t^2(\psi)$ than LOPI1, they have a better chance of finding the global minimum of STRESS. In chapter 4 we present a simulation study using the LOPI1, LOPI2, LOPI3, and LOPI4 strategies.

2.2.3 Simulated annealing

The method of simulated annealing has been proposed for combinatorial optimization by Kirkpatrick, Gelatt, and Vecchi (1983) and Černý (1985), see also Van Laarhoven and Aarts (1987). The idea of simulated annealing can be extended to general optimization problems. It earns its name from the cooling process (annealing) in solid state physics. The aim is to cool material down to its lowest energy state by slowly reducing the temperature. If an equilibrium is reached, the temperature is somewhat increased to allow defects frozen in the structure to become loose again, so that after cooling down, a lower energy state of the material can be reached. The simulation of this annealing process serves as a metaphor in optimization.

De Soete, Hubert, and Arabie (1988) applied simulated annealing to unidimensional scaling. Their procedure can be described as follows:

1. $\tau \leftarrow 2.5t^2(\psi^0)$, $k \leftarrow 0$, $\psi^{(k)} \leftarrow \psi^0$, $r \leftarrow 2n$
2. $\psi'' \leftarrow \psi^{(k)}$, $c \leftarrow 1$, $k \leftarrow k + 1$
3. Repeat r times
4. $\quad c \leftarrow 0$
4. $\quad$ Perturb ψ'' into ψ' by a random pairwise interchange
5. $\quad$ Set $\psi'' \leftarrow \psi'$ if $t^2(\psi') \geq t^2(\psi'')$ or

if $t^2(\psi') < t^2(\psi'')$ with probability $\exp([t^2(\psi') - t^2(\psi'')]/\tau)$

6. If $\psi'' = \psi'$ then $c \leftarrow 1$
7. End repeat
8. $\psi^{(k+1)} \leftarrow \psi''$, $\tau \leftarrow \tau/2$, $r \leftarrow 1.1r$
9. If a change occurred ($c = 1$) then go to 2

Every perturbation that increases $t^2(\psi)$ is always accepted. If it decreases $t^2(\psi)$, it is accepted with a probability that decreases during the process, so that eventually the system freezes. Evidently, changing the initial settings of the cooling scheme or changing the diminishing rule of the temperature τ also changes the behaviour of the algorithm.

De Soete, Hubert, and Arabie (1988) used simulated annealing on unidimensional scaling and concluded that it did not perform better than the LOPI4 procedure.

2.2.4 The tabu search

Here, we discuss a heuristic method heavily based on a local search procedure like the pairwise interchange that finds a local maximum fast and tries to find subsequently better local maxima. The tabu search was proposed by Glover (1989, 1990) and used by De Amorim, Barthélemy, and Ribeiro (1992) in the context of clustering. It consists of two steps, a local search step yielding a local maximum and a global step in which a permutation is sought with a higher or at least equal value of $t^2(\psi)$ and that is different from the current local maximum permutation. Once found, we return to the local step, and so on. In the global step some search directions leading to the current local maximum are forbidden, hence the name tabu search.

The local step of tabu search is the pairwise interchange strategy of section 2.2.2. In the global search we wish to find a permutation different from the one we found so far, that has the same maximum value, or even higher. From that point we can return to our local search again, and so on. During the global search we accept the permutation that gives the highest value of $t^2(\psi)$, unless ψ is one of the tabu permutations that lay in the path leading to a previously found local maximum. In the latter case we accept the next best solution that is not a tabu permutation. The global search differs from the local search by adding an acceptance rule. Thus, in the global step we are only willing to decrease the value of $t^2(.)$, if it avoids returning to the previous local maximum.

Sometimes the method is called a steepest ascent, mildest descent method. In the next chapter it will turn out that the tabu search may be regarded as a discrete version of the tunneling method. It can also be seen as a non-probabilistic form of simulated annealing. Only in a limiting case can a global maximum be guaranteed. Then all ascending paths leading to all local maxima are tabu, which clearly is of no practical use for large n. In chapter 4 we present a simulation study to compare the tabu search with other strategies.

Using the ingredients set out above, we can summarize the tabu search algorithm as follows:

1. Initialise
   Set $\psi^{(1)} \leftarrow \psi^0$, $k \leftarrow 0$.

2. Local step
   a. $k \leftarrow k + 1$.
   b. Regard pairwise interchanges of $\psi^{(k)}$ and let ψ^t be the permutation with the highest value of $t^2(.)$.
   c. If $t^2(\psi^t)$ has a lower value than $t^2(\psi^{(k)})$ then $\psi^{(k)}$ defines a local maximum and go to the global step 3a.
   d. Set $\psi^{(k+1)} \leftarrow \psi^t$ and go to step 2a.

3. Global step
   a. $\psi^l \leftarrow \psi^{(k)}$
   b. Regard all pairwise interchanges of ψ^l and let ψ^g be the permutation with the highest value of $t^2(.)$ that is not in the Tabu list.
   c. If $t^2(\psi^g)$ has a higher value than $t^2(\psi^{(k)})$ then set $\psi^{(k+1)}$ equal to ψ^g and go to local step 2a.
   d. If $t^2(\psi^g)$ has a lower value than $t^2(\psi^l)$ then add ψ^g to the Tabu list.
   e. Set $\psi^l \leftarrow \psi^g$. If the Tabu list gets longer than s points then remove the oldest tabu permutation.
   f. If we cycle in the global step for more than r iterations or we return to the previous local minimum, then we stop and declare $\psi^{(k)}$ to be the candidate global minimum.

The tabu search always wants to find a permutation that has a higher value of the function, unless it appears to be on a path leading to a previously found local maximum. In that case we accept a permutation with a lower function value, but we take one with the smallest possible decrease. If it would have been possible to use a tabu list as large as the total number of permutations, then this strategy would produce a global maximum with certainty. Obviously, even for small size n, this would be much more impractical than using the dynamic programming approach outlined in the previous section. By making the tabu list too short we may cycle through the permutations and return to a previously visited permutation. Another aspect determining the success of the global step is the maximum number of iterations after which we terminate the global search. If it is too tight, we may not find a permutation with a higher value than the local maximum one, although it may exist. If it is too loose the method becomes inefficient and may run into cycles.

There exists a nice relation with simulated annealing (SA). The main similarity between both methods is that they sometimes accept a step which decreases the function value. The difference between SA and the tabu search is the acceptance rule. The tabu

search accepts a decrease in function value if there exists no better solution in the neighbourhood that is not forbidden. SA always accepts a permutation that increases the function value. If it does not, then SA still may accept the permutation with a certain probability. This probability tends to zero to damp the process.

2.3 Space covering methods using Lipschitz constants

After having considered unidimensional scaling, we return to multidimensional scaling and focus on space covering methods in this section. A useful property in space covering methods is the Lipschitz constant. A function is said to have a Lipschitz constant L if it satisfies the inequality

$$|\sigma(\mathbf{X}_1) - \sigma(\mathbf{X}_2)| \leq L\|\mathbf{X}_1 - \mathbf{X}_2\|. \tag{2.5}$$

Often the Euclidean norm is used, but this may not be necessarily so. Suppose that (2.5) holds and that we have some sample configurations $\mathbf{X}_i$ with $\mathbf{X}^*$ equal to $\text{argmin}_i\, \sigma(\mathbf{X}_i)$. Then it is not difficult to show that all $\mathbf{X}$ within distance $(\sigma(\mathbf{X}_i) - \sigma(\mathbf{X}^*))/L$ of $\mathbf{X}_i$ necessarily have STRESS larger than $\sigma(\mathbf{X}^*)$.

The following theorem shows that STRESS has a Lipschitz constant.

Theorem 2.1
Using the Euclidean norm in the metric $\mathbf{V}$, STRESS has a Lipschitz constant 1.

Proof
Let $\boldsymbol{\delta}$, $\mathbf{d}_1$, and $\mathbf{d}_2$ be the vectors of length $n(n-1)/2$ with lower triangular elements of Δ, $\mathbf{D}(\mathbf{X}_1)$, and $\mathbf{D}(\mathbf{X}_2)$ respectively, so that $\sigma(\mathbf{X}_1) = \|\boldsymbol{\delta} - \mathbf{d}_1\|_{\mathbf{w}}$ with $\mathbf{w}$ a diagonal matrix with weights w_{ij}, and $\|\mathbf{d}_1 - \mathbf{d}_2\|_{\mathbf{w}}^2 = \sum_{i<j} w_{ij}(d_{ij}(\mathbf{X}_1) - d_{ij}(\mathbf{X}_1))^2$. From the triangle inequality we have

$$\sigma(\mathbf{X}_1) = \|\boldsymbol{\delta} - \mathbf{d}_1\|_{\mathbf{w}} \leq \|\boldsymbol{\delta} - \mathbf{d}_2\|_{\mathbf{w}} + \|\mathbf{d}_1 - \mathbf{d}_2\|_{\mathbf{w}}$$
$$\sigma(\mathbf{X}_2) = \|\boldsymbol{\delta} - \mathbf{d}_2\|_{\mathbf{w}} \leq \|\boldsymbol{\delta} - \mathbf{d}_1\|_{\mathbf{w}} + \|\mathbf{d}_1 - \mathbf{d}_2\|_{\mathbf{w}}$$

which makes

$$|\sigma(\mathbf{X}_1) - \sigma(\mathbf{X}_2)| \leq \|\mathbf{d}_1 - \mathbf{d}_2\|_{\mathbf{w}}.$$

The square of the right-hand term can be written as

$$\begin{aligned}\|\mathbf{d}_1 - \mathbf{d}_2\|_{\mathbf{w}}^2 &= \sum_{i<j} w_{ij}\, d_{ij}^2(\mathbf{X}_1) + \sum_{i<j} w_{ij}\, d_{ij}^2(\mathbf{X}_2) - 2\sum_{i<j} w_{ij}\, d_{ij}(\mathbf{X}_1) d_{ij}(\mathbf{X}_2) \\ &= \text{tr}\mathbf{X}_1'\mathbf{V}\mathbf{X}_1 + \text{tr}\mathbf{X}_2'\mathbf{V}\mathbf{X}_2 - 2\sum_{i<j} w_{ij}\, d_{ij}(\mathbf{X}_1) d_{ij}(\mathbf{X}_2).\end{aligned}$$

By the Cauchy-Schwarz inequality $-d_{ij}(\mathbf{X}_1)d_{ij}(\mathbf{X}_2) \leq -(\mathbf{x}_{1,i} - \mathbf{x}_{1,j})'(\mathbf{x}_{2,i} - \mathbf{x}_{2,j})$ we have

$$\|\mathbf{d}_1 - \mathbf{d}_2\|_w^2 \leq \text{tr}\mathbf{X}_1' \mathbf{V}\mathbf{X}_1 + \text{tr}\mathbf{X}_2' \mathbf{V}\mathbf{X}_2 - 2\text{tr}\mathbf{X}_1' \mathbf{V}\mathbf{X}_2 = \|\mathbf{X}_1 - \mathbf{X}_2\|_V^2.$$

Combining these inequalities gives

$$|\sigma(\mathbf{X}_1) - \sigma(\mathbf{X}_2)| \leq \|\mathbf{d}_1 - \mathbf{d}_2\|_w \leq \|\mathbf{X}_1 - \mathbf{X}_2\|_V,$$

which proves that STRESS has a Lipschitz constant of one. Q.E.D.

The Lipschitz constant is defined in the Euclidean norm with the metric $\mathbf{V}$, i.e., $\|\mathbf{X}_1 - \mathbf{X}_2\|_V^2 = \text{tr}(\mathbf{X}_1 - \mathbf{X}_2)' \mathbf{V}(\mathbf{X}_1 - \mathbf{X}_2)$. It means that if $\sigma(\mathbf{X}_1) \geq \sigma(\mathbf{X}_2)$ within an ellipsoid around $\mathbf{X}_1$ defined by $\mathbf{V}$ with length smaller than $\sigma(\mathbf{X}_1) - \sigma(\mathbf{X}_2)$, STRESS must be larger than or equal to $\sigma(\mathbf{X}_2)$. This theorem can easily be extended to the unweighted Euclidean norm, but then we get a less sharp Lipschitz constant. A disadvantage of not using ellipsoids is that the volume covered is smaller than the one using $\mathbf{V}$ (unless $\mathbf{V}$ has equal eigenvalues), since a sphere fitting inside an ellipsoid has a smaller volume than the ellipsoid. Let λ be the largest eigenvalue of $\mathbf{V}$. Then matrix $\mathbf{V} - \lambda\mathbf{I}$ is negative semi-definite, so that the inequality $\|\mathbf{X}_1 - \mathbf{X}_2\|_V^2 \leq \lambda\|\mathbf{X}_1 - \mathbf{X}_2\|^2$ must hold. The Lipschitz constant of STRESS in unweighted Euclidean norm equals $\lambda^{1/2}$ for column centered $\mathbf{X}$. If all weights are equal to one, then $\mathbf{V} = n\mathbf{I} - \mathbf{1}\mathbf{1}'$, which has n as its largest eigenvalue. In this case the Lipschitz constant in unweighted Euclidean norm is equal to $n^{1/2}$.

The Lipschitz constant is of particular use in space covering methods. In this way parts of the space can be excluded from containing the global minimum. One of the problems of this method is to cover the full space. Although the Lipschitz condition of STRESS is defined in terms of hyperball or ellipsoids, we can always find a hypercube that fits inside this area. In this way it is easier to cover the space with contiguous subsets. However, the space covering method suffers from the curse of dimensionality. Timmer (1984) shows that the computational effort required increases exponentially with np. Therefore, we shall not use space covering methods to minimize STRESS in its general form. However, this method may be used to obtain the global minimum of a related problem, finding the minimum of STRESS with all objects fixed, but one object free. Clearly, this does not have to lead to the global minimum of the STRESS function, but a stricter local minimum is obtained, i.e., a local minimum where each object has a global minimum conditioned on the other objects.

2.4 Multistart

The multistart algorithm is a simple but powerful method, often used to find a global minimum. It consists of repeatedly choosing different (random) start values, each followed by a local search procedure. Clearly, the local minimum with the lowest function

value is a candidate global minimum. The strength of the multistart method is its simplicity combined with attractive theoretical properties. When using random start values and under mild smoothness assumptions, it can be proven that the multistart method asymptotically yields the global optimum with probability 1. However, we need infinitely many start configurations and local searches to reach the asymptotics. Besides, multistart is wasteful because the same local minimum will be found by various start configurations that are within the *region of attraction* of the same local minimum. Boender (1984) developed stopping rules based on a Bayesian estimate of the number of local minima and the relative size of the region of attraction. If we assume that each number of local minima w is equally likely to occur and we assume that given w the relative sizes of the regions of attraction for each local minimum follow a uniform distribution on the $(w-1)$ dimensional surface, then the posterior expectation of the number of local minima is

$$\frac{w(L-1)}{L-w-2}. \tag{2.6}$$

where L is the number of local searches performed so far. A simple termination rule is to stop whenever the number of local minima found differs only 0.5 from the expected number of local minima.

A better performance is obtained if we would start a local search procedure only once for each region of attraction that belongs to a unique local minimum. This ideal is approximated with clustering algorithms, of which we discuss an attractive one in the next section.

2.5 Multi-Level-Single-Linkage clustering

The multi-level-single-linkage clustering algorithm (MLSL) for global optimization is a stochastic method with attractive theoretical properties developed by Timmer (1984) and Rinnooy Kan and Timmer (1987a,b). MLSL may be regarded as an improvement of multistart. Clearly, in most cases multistart is wasteful, because the same local minimum may be found many times. A better performance could be obtained if we were able to start a local search procedure only once for each region of attraction that belongs to a unique local minimum. This ideal is approximated by clustering algorithms. Various cluster methods are known in the global optimization literature, but according to Timmer (1984) (also, see Zhigljavsky, 1991, Törn and Žilinskas, 1989) an adapted version of single-linkage is an appealing one in the present context. Clustering is a technique that assigns points to clusters to find groups of similar points (Hartigan, 1975). In global optimization we regard each $n \times p = q$ configuration matrix $\mathbf{X}$ as a point in $\mathbf{R}^q$.

Each cluster (and therefore the region of attraction) is defined by a seed point (here a local minimum configuration) and all points that are within a critical distance r_k of each other. As r_k becomes smaller over the iterations, the clusters approximate the region of

attraction better. The single-linkage clustering algorithm is known for the sausage shaped clusters it finds, the so called chaining effect. For our purpose it is a useful effect, since the level sets L_y, given by $\{\mathbf{X} \in \mathbf{R}^q \mid \sigma(\mathbf{X}) \leq y\}$, may well have this form. The method is defined over a compact and convex subset S of $\mathbf{R}^q$, which is assumed to contain the global minimum. Here, we choose S to be a rectangular parallelepiped. The MLSL algorithm differs from ordinary single-linkage clustering in that it takes the function value of the sample points into account, hence the name multi-level-single-linkage. The algorithm can be summarized as follows

1. Initialise. Set $k \leftarrow 0$ and $X = \{\varnothing\}$.
2. (Global phase) Draw N initial configurations from the uniform distribution over S. Evaluate STRESS value for each configuration. Add the points to set X. Set $k \leftarrow k+1$. Adjust critical distance r_k.
3. (Local phase) Apply the local search procedure to every sample point $\mathbf{X}_i$ of X unless $\mathbf{X}_i$ is within a small distance ε of the boundary of S, or there exists another $\mathbf{X}_j$ with $\sigma(\mathbf{X}_j) < \sigma(\mathbf{X}_i)$ within critical distance r_k.
4. Decide by means of a stopping rule to stop or return to 2.

Note that other clustering methods can be applied by changing the selection rule in step 3. It is easy to see that it is not necessary to keep track of the clusters. The only information that is needed for the decision to start a local search procedure is derived from the distance between point $\mathbf{X}_i$ and the other points in the sample set X and their function values. MLSL depends critically on how the critical distance is defined. This should be done in such a way that eventually all local minima are found, including the global one, but not infinitely many local search procedures are needed. Timmer (1984) proved that these requirements are met by deriving the critical distance from volume measures and probabilities. For rigorous proofs of the method we refer to Timmer (1984) and Rinnooy Kan and Timmer (1987a,b). A simplified outline of the proof is presented here.

2.5.1 Why MLSL works

In developing global optimization algorithms two aspects are attractive. If the method is asymptotically correct, then the method guarantees that a global minimum will be found eventually. The second attractive aspect is that it should be efficient. We would like to find the global minimum within reasonable time, preferably before infinity. These two aspects can be proved for global optimization with MLSL, although for large n the efficiency declines.

Let $B_{\mathbf{A},r}$ be the set of points $\mathbf{X}$ within critical distance r of $\mathbf{A}$ with $\mathbf{A},\mathbf{X} \in S$ and let $A_{\mathbf{A},r}$ be the set of points in $B_{\mathbf{A},r}$ with $\sigma(\mathbf{X}) < \sigma(\mathbf{A})$. In the sequel, the critical distance in iteration k is denoted interchangeably by r or r_k. We want to make sure that for any new sample point $\mathbf{A}$ with increasing k the number of expected points in $A_{\mathbf{A},r}$ increases, so that

eventually no local searches are performed. On the other hand, r should become arbitrarily small so that no local minima are overlooked.

Suppose we have kN sample points from a uniform distribution where **A** is one of them. The probability that none of the other points is within distance r of **A** with smaller function value is

$$\left(1-\frac{\mathrm{m}(A_{\mathbf{A},r})}{\mathrm{m}(S)}\right)^{kN-1}, \tag{2.7}$$

where $\mathrm{m}(S)$ is the Lebesque measure of S, a volume measure. We shall see that MLSL is defined in such a way that this probability tends to zero after a sufficient number of iterations. This implies that the probability of finding another point within distance r with lower function value goes to one, so that eventually no local search is started. However, given r we do not know a priori $\mathrm{m}(A_{\mathbf{A},r})$. It turns out that a simple relationship exists between $\mathrm{m}(A_{\mathbf{A},r})$ and $\mathrm{m}(B_{\mathbf{A},r})$. It can be proven that as r decreases to 0, it holds that $\mathrm{m}(A_{\mathbf{A},r}) \geq \beta \mathrm{m}(B_{\mathbf{A},r})$ with $0 < \beta < 1/2$ (see Timmer, 1984; Rinnooy Kan and Timmer, 1987a,b). This can be seen intuitively for one dimension by looking at Figure 2.2. For large r, $\mathrm{m}(A_{\mathbf{A},r})$ can be larger than $1/2\ \mathrm{m}(B_{\mathbf{A},r})$, as in the figure or smaller than $1/2\ \mathrm{m}(B_{\mathbf{A},r})$. When zooming in, i.e., choosing r smaller and smaller, the surface of the function is approximated by a hyperplane. This holds for every distance measure that is symmetric around each of the q dimensions of **X**, for **X** not a stationary point. The points within distance r of **A** tend to be split in two equal parts, one with function values larger than $\sigma(\mathbf{A})$, the other half with smaller function values. Theorem 4.2.2.1 of Timmer (1984) implies that we can always find a β, with $0 < \beta < 1/2$ (if the function is twice continuously differentiable and **A** is not a stationary point) such that $\mathrm{m}(A_{\mathbf{A},r}) \geq \beta \mathrm{m}(B_{\mathbf{A},r})$ holds for r smaller than some small constant.

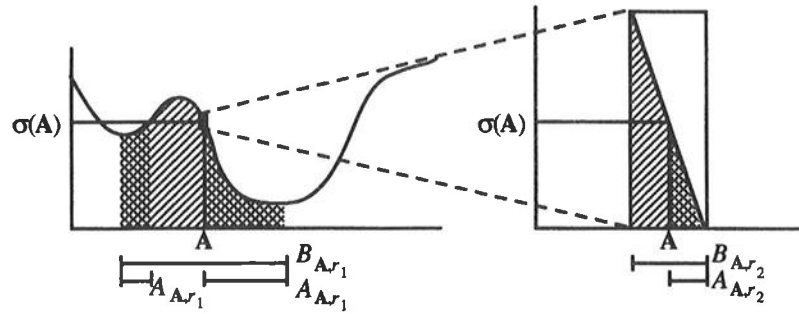


Figure 2.2 *For decreasing* r_k, $\mathrm{m}(A_{\mathbf{A},r})$ *gets larger than* $\beta\ \mathrm{m}(B_{\mathbf{A},r})$ *for some* $0 < \beta < 1/2$.

Using this result, an upper bound can be found of the probability that there is no point in the sample set within distance r_k with smaller function value (2.7), i.e.,

$$
\begin{aligned}
\mathrm{m}(A_{\mathrm{A},r}) &\geq \beta\,\mathrm{m}(B_{\mathrm{A},r}),\\
-\frac{\mathrm{m}(A_{\mathrm{A},r})}{\mathrm{m}(S)} &\leq -\frac{\beta\,\mathrm{m}(B_{\mathrm{A},r})}{\mathrm{m}(S)},\\
\left(1-\frac{\mathrm{m}(A_{\mathrm{A},r})}{\mathrm{m}(S)}\right) &\leq \left(1-\frac{\beta\,\mathrm{m}(B_{\mathrm{A},r})}{\mathrm{m}(S)}\right), \qquad (2.8)\\
\left(1-\frac{\mathrm{m}(A_{\mathrm{A},r})}{\mathrm{m}(S)}\right)^{kN-1} &\leq \left(1-\frac{\beta\,\mathrm{m}(B_{\mathrm{A},r})}{\mathrm{m}(S)}\right)^{kN-1}.
\end{aligned}
$$

The upper bound of this probability depends directly on $\mathrm{m}(B_{\mathrm{A},r})$, which in turn depends directly on r_k for properly chosen distance measures. For r_k smaller than some small constant, (2.8) shows that it is expected to find other points within distance r_k, let alone points with smaller function values, eventually with probability 1. The decision rule for starting a local search procedure from **A** implies that no local search is started. Thus eventually no local searches shall be performed.

Suppose that $B_{\mathrm{A},r}$ is entirely contained in S. Then the expected number of sample points in $B_{\mathrm{A},r}$ is $kN\mathrm{m}(B_{\mathrm{A},r})/\mathrm{m}(S)$. By setting this equal to $\gamma \ln kN$, where γ is a pre-set positive constant and ln is the natural logarithm, the density of points in $B_{\mathrm{A},r}$ gets higher with increasing number of iterations. As r becomes smaller, the number of points within distance r of **A** increases, and so does the number of points within distance r with smaller STRESS values. Consequently, the probability that a local search is started from **A** goes to zero.

As an intermezzo, we could use the arguments outlined above to construct a stochastic form of a space covering method. The sampling method and definition of the critical distance could be used from MLSL. Remember that covering methods do not have local searches. Then we know that the density of sample points within distance r of **A** increases over the iterations, whereas r decreases. Due to the fact that STRESS has a Lipschitz constant, we get increasingly better lower bounds of STRESS. Furthermore, the stochastic process ensures that the entire feasible region gets covered. After termination, we end up with an upper and lower bound of the global minimum of STRESS that is directly dependent on the Lipschitz constant of STRESS and the critical distance r. Unfortunately, to obtain an acceptable range for the global minimum we need a huge number of sample points, which increases drastically with the dimensionality q. We shall not discuss this further.

Clearly, some stopping rule is needed for the MLSL algorithm. Fortunately, the assumptions of Boender (1984) for terminating the multistart method are still valid, thus we may use the same stopping rule. However, for large scale problems we may have to stop earlier because of problems with the storage of all the sample points. It remains to be shown how r_k can be derived for a proper distance measure. Before discussing this in section 2.5.3, we first investigate the feasible region S.

2.5.2 The feasible region S

The MLSL method requires the specification of a feasible region S. We present two ways of determining S. First of all, we have to determine the number of parameters of the $n \times p$ configuration matrix $\mathbf{X}$. A configuration matrix determines the value of STRESS uniquely up to an arbitrary translation, rotation or reflection. Thus one local minimum configuration $\mathbf{X}^*$ generates infinitely many other local minima, simply by using the above mentioned transformations. Since the MLSL method searches for all local minima this is an undesirable feature which can be solved by constraining the configuration matrix. A second way of solving this problem is by using distances, which is discussed later on. Using constraints on the configuration, the translation invariance can be solved by fixing one point of the configuration matrix $\mathbf{X}$ at the origin, say the first row $\mathbf{x}_1'$. The rotation invariance is constrained by letting the second row $\mathbf{x}_2'$ only vary along the first axis and setting the remaining coordinates fixed to zero. For p larger than 2 we free the coordinates of the third row only on the first two axes and fixing the rest to zero, etc. Thus a configuration matrix $\mathbf{X}$ with 3 dimensions is constrained to be

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 \\ x_{21} & 0 & 0 \\ x_{31} & x_{32} & 0 \\ x_{41} & x_{42} & x_{43} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n3} \end{bmatrix}. \tag{2.9}$$

The reflection problem is solved by restricting the first subdiagonal elements (x_{21}, x_{32} and x_{43} in the 3 dimensions example above) to be nonnegative. The constraints reduce the number of free parameters to $q = (n-1)p - p(p-1)/2$.

From any unconstrained $\mathbf{X}$ the constrained matrix $\mathbf{X}^+$ may be obtained as follows. First the origin is set to the first row $\mathbf{x}_1'$ by $\mathbf{X}^+ \leftarrow \mathbf{X} - \mathbf{1}\mathbf{x}_1'$. Thus $\mathbf{x}^+_1$ equals $\mathbf{0}$. Further, let $\mathbf{X}_{(p)}$ be the $p \times p$ matrix of the rows $\mathbf{x}^+_2$ to $\mathbf{x}^+_{p+1}$ and $\mathbf{T}$ be the modified Gram-Schmidt decomposition of $\mathbf{X}_{(p)}'$ which makes $\mathbf{T}$ a rotation matrix with $\mathbf{T}'\mathbf{T} = \mathbf{I}$ and $\mathbf{X}_{(p)}' = \mathbf{T}\mathbf{L}'$ with $\mathbf{L}$ a lower triangular matrix. Note that the diagonal elements of $\mathbf{L}$ are always positive. Then $\mathbf{X}_{(p)}\mathbf{T} = \mathbf{L}$ and thus $\mathbf{X}^+\mathbf{T}$ is of the form (2.9), meeting the desired constraints.

We could define the feasible region S to be a large $(n-1)p - p(p-1)/2$ dimensional hypercube, but a sharper set exists. We can find it by making use of a known configuration $\mathbf{X}$ with a preferably low STRESS value. A sharp bound could be obtained by applying classical scaling followed by SMACOF, which gives us the first local minimum. The feasible region S can be determined by examining the definition of STRESS closer. Clearly it holds that

$$w_{ij}(\delta_{ij} - d_{ij}(\mathbf{X}))^2 \leq \sigma^2(\mathbf{X}), \tag{2.10}$$

since STRESS is the sum of $w_{ij}(\delta_{ij} - d_{ij}(\mathbf{X}))^2$ over $i<j$. This implies

$$\max(\delta_{ij} - w_{ij}^{-1/2}\sigma(\mathbf{X}), 0) \leq d_{ij}(\mathbf{X}) \leq \delta_{ij} + w_{ij}^{-1/2}\sigma(\mathbf{X}), \tag{2.11}$$

and therefore point j must be located inside a p-dimensional hypersphere with centre point i and radius $\delta_{ij} + w_{ij}^{-1/2}\sigma(\mathbf{X})$, but outside a hypersphere with radius $\max(\delta_{ij} - w_{ij}^{-1/2}\sigma(\mathbf{X}), 0)$. In the sequel we shall not make use of the latter lower bound unless indicated otherwise. For each pair of points, one of which is fixed, we may find the radius of the p-dimensional hypersphere in which the point must be located. Given point j the coordinate x_{is} of point i must be in the interval $[-\delta_{ij} - w_{ij}^{-1/2}\sigma(\mathbf{X}) + x_{is}, \delta_{ij} + w_{ij}^{-1/2}\sigma(\mathbf{X}) + x_{is}]$. Let the lower bound of x_{is} be represented by $\underline{x}_{is}$ and the upper bound by $\bar{x}_{is}$, so that the interval becomes $[\underline{x}_{is}, \bar{x}_{is},]$. The upper bound of $\bar{x}_{is}$ is defined $x_{is} + w_{ij}^{-1/2}\sigma(\mathbf{X}) + \max_{j\neq i}\delta_{ij}$, the lower bound $\underline{x}_{is}$ by $x_{is} - w_{ij}^{-1/2}\sigma(\mathbf{X}) - \max_{j\neq i}\delta_{ij}$. Note that the intervals on all p dimensions define the smallest hypercube that contains the hypersphere with radius $\delta_{ij} + w_{ij}^{-1/2}\sigma(\mathbf{X})$. Although the hypercube has a larger volume than the hypersphere it has the advantage that it is easier to handle.

Above it was explained how a unique configuration, without freedom of rotation translation and reflection, is obtained and how intervals for the coordinates may be obtained. However, nothing has been said about which point should be fixed and what the order of the points should be, as long as the coordinates satisfy the restrictions of (2.9). We still may exploit this freedom by changing the order of the objects as to minimize the Lebesque measure and thus reduce the size of S. Clearly, changing the order of the points does not matter as long as this order is consistently used throughout the method. Here we set the origin in the point that reduces S the most, i.e., the point with the largest interval. Since its coordinates are fixed to zero, no contribution is given to the Lebesque measure of S. A further reduction of $m(S)$ is obtained when we place the point which has the second largest interval in second position. Since this point has only positive values on the first axis and zero values on the second, no extra contribution is given to $m(S)$. Instead of a p-dimensional hypercube only a 1-dimensional interval is given. Clearly, this reduces the volume of S too. The same procedure is followed to select further points up to the p-th position.

A second way of getting rid of the rotation, translation and reflection invariance is by using the distance between the vectors of distances $\mathbf{d}$. Clearly, both $\mathbf{X}$ and its rotation have the same vector of distances. The two decisions that are to be made by MLSL are: a. for $\sigma(\mathbf{X}_2) \leq \sigma(\mathbf{X}_1)$ is $\mathbf{X}_1$ outside critical distance of $\mathbf{X}_2$ so that a local search should be started from $\mathbf{X}_1$, and b. once a local search is started, is the local minimum just found different from other local minima already found. From the proof of theorem 2.1 we know that $\|\mathbf{d}_1 - \mathbf{d}_2\|_w \leq \|\mathbf{X}_1 - \mathbf{X}_2\|_V$. Therefore, if $\|\mathbf{d}_1 - \mathbf{d}_2\|_w \geq r$, then also $\|\mathbf{X}_1 - \mathbf{X}_2\|_V \geq r$. This means that if we decide not to start a local search on basis of the distance vectors, we would not have started a local search if we based our decision on $\|\mathbf{X}_1 - \mathbf{X}_2\|_V$. Conversely, it might happen that a local search is started, although the decision rule based on $\|\mathbf{X}_1 - \mathbf{X}_2\|_V$ would not start it.

Let us now return to the second demarcation option of S. It can be obtained if we have a good estimate of the lowest local minimum found so far, say $\sigma(\mathbf{X}^*)$. Since at a stationary point $\eta^2(\mathbf{X}^*) = \rho(\mathbf{X}^*)$, we may write $\sigma^2(\mathbf{X}^*) = \eta_\delta^2 - \eta^2(\mathbf{X}^*)$. Thus, if we wish to find a local minimum with $\sigma^2(\mathbf{X}) < \sigma^2(\mathbf{X}^*)$, then $\eta^2(\mathbf{X})$ must be larger than $\eta^2(\mathbf{X}^*)$ and is always smaller than (or equal to) η_δ^2. This means that the global minimum must always be outside a hypersphere of radius $\eta(\mathbf{X}^*)$, but inside a hypersphere of length η_δ, i.e., inside the peel defined by the two spheres. Thus we may adapt S during the MLSL iterations so that S becomes smaller as sharper bounds of the global minimum are known. One problem that remains is that we have to draw sample points that are uniformly distributed within a hyperball. This can be done by using generalized polar coordinates. Let the $q = np$ vector $\mathbf{x}$ denote vec($\mathbf{X}$), that is the columns of $\mathbf{X}$ placed underneath each other. The generalized polar coordinates of $\mathbf{x}$ are given by

$$\begin{aligned}
x_1 &= r \sin\theta_1 \\
x_2 &= r \cos\theta_1 \sin\theta_2 \\
x_3 &= r \cos\theta_1 \cos\theta_2 \sin\theta_3 \\
&\vdots \\
x_{q-1} &= r \cos\theta_1 \cos\theta_2 \cos\theta_3 \ldots \cos\theta_{q-2} \sin\theta_{q-1} \\
x_q &= r \cos\theta_1 \cos\theta_2 \cos\theta_3 \ldots \cos\theta_{q-2} \cos\theta_{q-1},
\end{aligned} \tag{2.12}$$

where $r \geq 0$, $0 \leq \theta_1 \leq 2\pi$, and $-\pi \leq \theta_i \leq \pi$ for $i = 1, \ldots, q-1$. It can be easily verified that indeed $\sum_i x_i^2 = r^2$. If r is kept fixed, then a uniform distribution on the hyperball with radius r is obtained by drawing θ_i uniformly from the appropriate interval. For the density to be constant within the hyperball S it must be true that the probability of finding a sample point $\mathbf{A}$ within radius r inside the hyperball S is equal to

$$\frac{\mathrm{m}(B_{\mathbf{A},r})}{\mathrm{m}(S)} = \frac{\pi^{q/2} r^q}{\mathrm{m}(S)\Gamma(1+q/2)} = c \tag{2.13}$$

where $\Gamma(x)$ is the gamma function. This ratio must be equal for all radii. The only way to assure this is by requiring that r^q is uniformly distributed, thus by drawing an r' from the uniform distribution between 0 and the $\eta_\delta^{1/q}$ and setting $r = r'^{1/q}$. If we want to draw a sample point from the uniform distribution in the peel defined above, then it suffices to draw r' from the uniform distribution between $\eta^{1/q}(\mathbf{X}^*)$ and $\eta_\delta^{1/q}$.

We presented two ways of finding a demarcation of S, the first one defining a parallelepiped, the second one an ellipsoid. For now it remains open which one is to be used, but a guideline could be to use whichever one has the smallest Lebesque measure.

2.5.3 *Definitions of the critical distance*

Up to now the set $B_{\mathbf{A},r}$ is defined to be the set of all points within critical distance r of $\mathbf{A}$, but we did not specify the distance norm of the critical distance. Two distance norms seem reasonable, the Euclidean distance defining hyperballs and the max norm defining hypercubes. Timmer (1984) used the Euclidean norm $\|\mathbf{A} - \mathbf{X}\|$ originally in MLSL. Then $B_{\mathbf{A},r}$ defines a hyperball with volume

$$\mathrm{m}(B_{\mathbf{A},r}) = \frac{\pi^{q/2} r^q}{\Gamma(1+q/2)}. \tag{2.14}$$

Since all kN sample points are uniformly distributed, the probability that there is a sample point within distance r of $\mathbf{A}$ is $\mathrm{m}(B_{\mathbf{A},r})/\mathrm{m}(S)$ if $B_{\mathbf{A},r}$ is entirely contained in S. In section 2.5.1 it was assumed that this ratio equals $\gamma \ln kN / kN$. From the right part of

$$\frac{kN}{\mathrm{m}(S)} \mathrm{m}(B_{\mathbf{A},r}) = \frac{kN}{\mathrm{m}(S)} \frac{\pi^{q/2} r^q}{\Gamma(1+q/2)} = \gamma \ln kN \tag{2.15}$$

we derive the critical distance

$$r_k = \pi^{-1/2} \left(\Gamma(1+q/2)\, \mathrm{m}(S) \frac{\gamma \ln kN}{kN} \right)^{1/q}. \tag{2.16}$$

It can be proven that for $\gamma > 4$ the number of local searches started is finite with probability 1, even if the sampling continues for ever (Timmer, 1984; Rinnooy Kan and Timmer, 1987a,b).

The second option for choosing the definition for the critical distance is the max norm, or the L_∞ norm. With this option, the set $B_{\mathbf{A},r}$ defines a hypercube with midpoint $\mathbf{A}$ which has distance r to all sides, and volume

$$\mathrm{m}(B_{\mathbf{A},r}) = (2r)^q. \tag{2.17}$$

Setting $\mathrm{m}(B_{\mathbf{A},r})/\mathrm{m}(S)$ equal to $\gamma \ln kN / kN$ allows the critical distance to be expressed as

$$r_k = \frac{1}{2} \left(\frac{\gamma \ln kN}{kN} \right)^{1/q}. \tag{2.18}$$

In deriving (2.16) and (2.18) we assumed that $B_{\mathbf{A},r}$ is entirely contained in S. In the next section we shall see that for most sample points, $B_{\mathbf{A},r}$ is *not* entirely contained in S when r is relatively large. Since r is large during the initial stage, a local search will be started from almost every sample point.

2.5.4 *The curse of dimensionality*

One problem with the current formulation of MLSL for STRESS is its lack of efficiency for high dimensionality q. Especially during the initial stage, the probability of starting a local search from a sample point is very large. This is caused by the fact that in the beginning the number of sample points is small and therefore r is large. However, if there are not so many sample points then the probability of a sample point being near the boundary of S is also very large. This can be seen from either of the two definitions of S. Suppose that S is a sphere. Then, from the transformation needed for the radius of the generalized polar coordinates, we see that the majority of sample points are near the border of S. Now let S be a hypercube. Then too it is not difficult to show that most sample points are located near the boundary of S. It can be seen when comparing the contents hypercube with the contents of a hyperball that fits exactly in the cube. For higher dimensions the difference between the two grows rapidly. This difference,

$$\left(2^q - \frac{\pi^{-q/2}}{\Gamma(1+q/2)}\right) r^q \,, \tag{2.19}$$

rapidly tends to $(2r)^q$ as q gets larger. Therefore, the only way to assure $kN\ \mathrm{m}(B_{\mathbf{A},r})/\mathrm{m}(S) = \gamma \log kN$ is to have large r. But for large r much of the hyperball around a given point **A** is located outside S. Consequently, if the critical distance is still large, much of the hyperball $B_{\mathbf{A},r}$ around sample point **A** is outside S. Thus the probability of finding another sample point within critical distance is (much) less then $kN\mathrm{m}(B_{\mathbf{A},r})/\mathrm{m}(S)$. Here again we encounter the curse of dimensionality.

To increase the efficiency of MLSL we follow Groenen (1992) who proposed to move $B_{\mathbf{A},r}$ with center point **C** such that **C** is as close as possible to **A** and that $B_{\mathbf{C},r}$ fits entirely inside S. If r is defined to be the Euclidean distance, then the hyperball $B_{\mathbf{A},r}$ has to be moved into S. However, if S is a hypercube then it may happen that **A** falls outside $B_{\mathbf{C},r}$, which would imply a local search anyhow from **A**. This problem is avoided when using the hyperball definition of S. On the other hand, if we are using the L_∞ norm then $B_{\mathbf{A},r}$ defines a hypercube that has to be moved such that it falls exactly inside S. Now we may have a problem again if S is a hypersphere, since it can happen that **A** falls outside $B_{\mathbf{C},r}$. Therefore, if r is defined in the L_∞ norm, the hypercube definition of S should be used.

Since the moving $B_{\mathbf{A},r}$ ensures that it always fits inside S, the expected number of sample points within critical distance r of each other equals indeed $\gamma \ln kN / kN$. Changing the MLSL algorithm this way does not change the other theoretical properties. When r is smaller than ε, no local search was started anyway, and thus no set $B_{\mathbf{A},r}$ has to be moved and the old situation applies.

To get an idea of the size of the critical distance given the number of points, r was computed for different numbers of sample points. We took S to be a unit hypercube and defined r in the L_∞ norm. In Figure 2.3 two typical curves are given, one for $n = 17$ and p

= 2, the other for $n = 9$ and $p = 2$. It can be seen that in the first example one needs about 10^{40} sample points to obtain a r smaller than 0.05 . This is a clear indication of the curse of dimensionality.

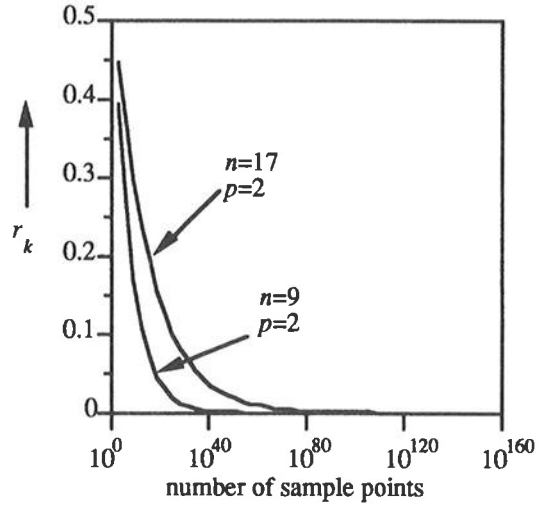


Figure 2.3 *Number of sample points and the corresponding critical distance where* S *is a unit hypercube for two small sized* MDS *problems.*

2.6 Concluding remarks

In this chapter we presented the problem of global optimization of STRESS. A classification of global optimization techniques was given. Some of them were discussed in more detail and applied to the STRESS function. We have split the scaling problem into three parts: full-dimensional scaling, unidimensional scaling and MDS for $1 < p < n - 1$. In the previous chapter it was shown that the full-dimensional scaling problem has a local minimum that is also global. Unidimensional scaling turned out to be a combinatorial problem, which can be solved globally by dynamic programming, or solved locally by using heuristic strategies like pairwise interchange, simulated annealing and the tabu search. For MDS with $1 < p < n - 1$ we discussed several techniques like space covering methods, multistart and multi-level-single-linkage clustering. We proved that STRESS has a Lipschitz constant. In the next chapter, we discuss and extend the tunneling method in great detail. In chapter 4, we present a simulation study that compares some of these global optimization methods in various settings.

CHAPTER 3

THE TUNNELING METHOD

In this chapter we study systematic ways for finding a decreasing series of local minima of the MDS STRESS function by using the tunneling method. This method consists of an iterative two-step procedure: in the first step a local minimum is sought and in the second one another configuration is determined with exactly the same STRESS. It can be described by the following analogy. Suppose we wish to find the lowest spot in a selected area in the Alps. First we have a local search where we pour some water and see where it stops. From this point we perform a global search by digging tunnels horizontally until we come out of the mountain. Then we pour water again and dig tunnels again. If we stay underground for a long time while digging the tunnel, we simply conclude that the last spot was in fact the lowest place in the area.

The tunneling method for functions of more parameters is mainly due to Montalvo (1979), Gomez and Levy (1982) and Levy and Gomez (1985). Earlier Vilkov, Zhidkov, and Shchedrin (1975) presented a one-parameter tunneling function. An important and attractive feature of the tunneling algorithm is that successive local minima always have lower or equal function values. The method can be classified among the generalized descent methods that use a penalty function. The tunneling method can also be viewed as a continuous version of the discrete tabu search discussed before, since both methods try to avoid previously found local minima and continue their search for other configurations with equal or lower function values.

The tunneling step is the crux of the tunneling method. It is performed by minimization of a particular function, called the tunneling function. To find another configuration with the same STRESS, this function must have several characteristics. Some of these characteristics are met by the tunneling function originally defined by Gomez and Levy (1982):

$$\tau_0(\mathbf{X}) = \frac{\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)}{\left\|\mathbf{X} - \mathbf{X}^*\right\|^2} \tag{3.1}$$

where $\mathbf{X}^*$ is the local minimum configuration. The first characteristic that (3.1) exhibits, is that it has zero points for configurations with STRESS equal to $\sigma(\mathbf{X}^*)$. Secondly, these zero points are not necessarily the lowest possible value of the tunneling function. Thirdly, the factor $\|\mathbf{X}^* - \mathbf{X}\|^{-2}$, also called the *pole* of the tunneling function, is used to create

elevated values of the tunneling function near the local minimum configuration $\mathbf{X}^*$, so that a zero point at $\mathbf{X}^*$ is presumably excluded. Because the pole is a factor, it does not change the positions of the zero points different from $\mathbf{X}^*$. Note that $\tau_0(\mathbf{X})$ is not defined at the local minimum $\mathbf{X}^*$, so that any minimization algorithm should start from somewhere else.

The complex task of finding the global minimum of the STRESS function has been replaced by a similarly complex problem, i.e., finding the zero points of the tunneling function by minimization . Fortunately, the latter problem has the distinct advantage of having an additional feature for a desired local minimum of the tunneling function: it should have STRESS smaller than or equal to $\sigma(\mathbf{X}^*)$. Clearly, if $\mathbf{X}^*$ is a unique global minimum it will be impossible to find a zero point of $\tau_0(\mathbf{X})$. However, for the moment we assume that $\mathbf{X}^*$ is not a unique global minimum, so that zero points of the tunneling function do exist.

The tunneling function (3.1) has some major defects, some of which are solely due to the STRESS function. A redefinition of the tunneling function that solves these defects is presented in section 3.1. We also explain why these problems may occur and give some of their mathematical properties. Section 3.2 shows how the redefined function can be minimized to obtain its zero points. Two important methods used for the minimization of the tunneling function are *iterative majorization* and *parametric programming*, which keeps tunneling within the majorization framework of SMACOF. These sections are based on Groenen and Heiser (1991). It may be necessary to have more than one pole in the tunneling function for the tunneling step to succeed. Therefore, we discuss the minimization of the tunneling function with more than one pole in section 3.3. In section 3.4 some applications of the tunneling method are given. Finally, in section 3.5 we experiment with various parameter settings of the tunneling algorithm for a good tuning of the method.

3.1 Redefining the tunneling function

Definition (3.1) of the tunneling function suffers from some problems that will be described shortly. In this section we first present a redefinition of the tunneling function that solves these problems. Then we explain in the next three subsections how the redefinition is built step by step and give the mathematical properties in greater detail.

The first problem involves changes of the tunneling function value when a configuration is rotated. This is inconsistent with the STRESS function, which is invariant under rotation. Therefore, the tunneling function has to be invariant under rotations too. The second problem involves the strength of the pole. If the pole is not strong enough, then the tunneling function has an undesired zero point at $\mathbf{X}^*$. It is shown analytically that a pole strength parameter must be included. A similar analytical result is obtained for the third problem that we will call attraction to the horizon. It turns out that we must be careful not to end up with very large configurations that also yield tunneling function

values close to zero. We now present a redefinition of the tunneling function that solves these problems and still exhibits the characteristics initially set out as desirable:

$$\tau_1(\mathbf{X}) = \left(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)\right)^{2\lambda}\left(1 + \frac{1}{\|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_w^2}\right)$$

$$= \left(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)\right)^{2\lambda}\left(\frac{\|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_w^2 + 1}{\|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_w^2}\right). \qquad (3.2)$$

where $\|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_w^2 = \sum_{i<j} w_{ij}(d_{ij}(\mathbf{X}^*) - d_{ij}(\mathbf{X}))^2$. The function $\tau_1(\mathbf{X})$ seems to satisfy all requirements: the zero points (that is, points with STRESS equal to $\sigma(\mathbf{X}^*)$) do not change after multiplication, a configuration that is a rotation of $\mathbf{X}$ yields the same tunneling function value, the pole can be made strong enough by adjusting the pole strength parameter λ ($0 < \lambda < 1$), and for large configurations minimizing $\tau_1(\mathbf{X})$ amounts to minimizing $(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*))^{2\lambda}$ because $1 + \|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_w^{-2}$ tends to one. A one dimensional slice of $\tau_1(\mathbf{X})$ is given in Figure 3.1. A summary of all the properties of (3.2) is given in Table 3.1.

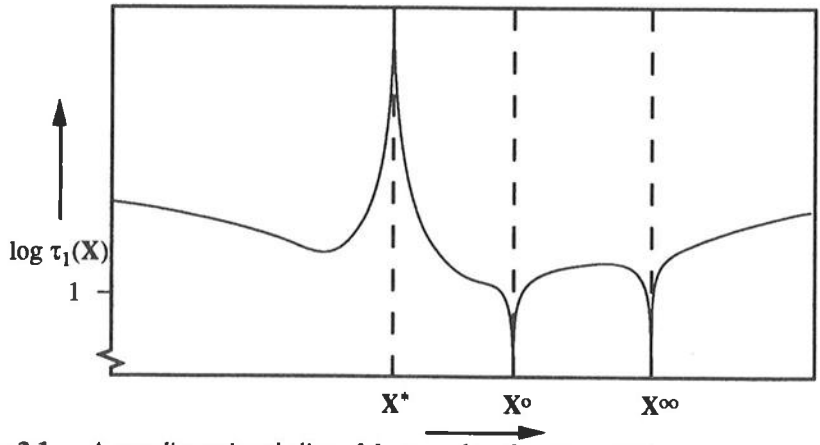


Figure 3.1 *A one dimensional slice of the tunneling function* $\tau_1(\mathbf{X})$.

In the next three subsections we derive the redefined tunneling function (3.2) in steps from the original one (3.1). In fact we start from

$$\tau_2(\mathbf{X}) = \frac{\left(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)\right)^2}{\left\|\mathbf{X} - \mathbf{X}^*\right\|^2}. \tag{3.3}$$

which is always nonnegative. Consequently, the zero points are the lowest value possible of the tunneling function.

Table 3.1 *The elements of the tunneling function* $\tau_1(\mathbf{X})$ *and their purpose.*

	Purpose	Element
1.	Zero point if STRESS is equal to local minimum STRESS.	$\tau(\mathbf{X}) = \sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)$
2.	Zero points are the lowest tunneling function values.	$\tau(\mathbf{X}) = \left(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)\right)^2$
3.	Avoid a zero point at $\mathbf{X}^*$ by erecting a pole.	$\tau(\mathbf{X}) = \left(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)\right)^2 / P(\mathbf{X})$
4.	Avoid a zero point at irrelevant transformations of $\mathbf{X}^*$.	$P(\mathbf{X}) = \left\|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\right\|_w^2$
5.	Ensure sufficiently strong pole (use pole strength parameter λ, $0 \le \lambda \le 1$).	$\tau(\mathbf{X}) = \left(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)\right)^{2\lambda} / P(\mathbf{X})$
6.	Avoid attraction to the horizon.	$\tau_1(\mathbf{X}) = \left(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)\right)^{2\lambda} (1 + 1/P(\mathbf{X}))$

3.1.1 Obtaining rotational invariance of the denominator

One basic property of the STRESS function is its invariance under rotation of the solution **X**. This property can be easily understood by realizing that the STRESS function is defined on the distances between points of the configuration, not on the configuration itself. Clearly, distances do not change under rotation, reflection, or translation. Therefore, it is desirable that this property is maintained in the tunneling function. Unreported numerical experiments showed that minimizing $\tau_2(\mathbf{X})$ leads to a rotation of the local minimum configuration within a small number of iterations. The behaviour of $\tau_2(\mathbf{X})$ can be understood by examining it more closely. The problem is caused by the numerator $(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*))^2$, since the STRESS function is invariant under rotation of **X**. It can be seen quite easily that a rotation of $\mathbf{X}^*$ is a minimum of the tunneling function. Let **R** be a rotation matrix with $\mathbf{R}'\mathbf{R} = \mathbf{R}\mathbf{R}' = \mathbf{I}$. Since $\sigma(\mathbf{X}^*\mathbf{R}) = \sigma(\mathbf{X}^*)$, the numerator of $\tau_2(\mathbf{X}^*\mathbf{R})$ is zero. Furthermore, the denominator is

$$\|\mathbf{X}^*\mathbf{R} - \mathbf{X}^*\|^2 = \text{tr}\, \mathbf{X}^*(2\mathbf{I} - \mathbf{R} - \mathbf{R}')\mathbf{X}^{*\prime} \geq 0. \tag{3.4}$$

For the left-hand-side to be strictly greater than zero, it is sufficient that $2\mathbf{I} - \mathbf{R} - \mathbf{R}'$ is positive definite. The latter can be verified by using the Cauchy-Schwarz inequality twice

$$\|\mathbf{Y}\|\,\|\mathbf{YR}\| = \text{tr}\, \mathbf{YY}' \geq \text{tr}\, \mathbf{YRY}'$$
$$\|\mathbf{Y}\|\,\|\mathbf{YR}'\| = \text{tr}\, \mathbf{YY}' \geq \text{tr}\, \mathbf{YR}'\mathbf{Y}'$$

for any rotation matrix $\mathbf{R}$, with equality if and only if $\mathbf{R} = \mathbf{I}$. Thus, any $\mathbf{R} \neq \mathbf{I}$ yields a non-zero denominator. The zero numerator and non-zero denominator of $\tau_2(\mathbf{X}^*\mathbf{R})$ imply a minimum, but not a desirable one. Therefore, we need to adapt the denominator to remove the possibility of finding rotations of the local minimum configuration.

This can be achieved by applying the very same idea that makes the STRESS function invariant to rotations of the configuration. We simply use a function that measures the difference between the distances of the local minimum configuration and the distances of the current configuration. Or, to put it differently, we take the squared norm of the difference between the distance matrix of $\mathbf{X}^*$ and the distance matrix of $\mathbf{X}$, i.e.,

$$\left\|\mathbf{D}(\mathbf{X}) - \mathbf{D}(\mathbf{X}^*)\right\|_{\mathbf{w}}^2 = \left\|\mathbf{d} - \mathbf{d}^*\right\|_{\mathbf{w}}^2 = \sum_{i<j}^{n} w_{ij}\left(d_{ij}(\mathbf{X}) - d_{ij}(\mathbf{X}^*)\right)^2. \tag{3.5}$$

This denominator can be interpreted as the squared distance between the vector of distances of the local minimum configuration and the vector of distances among rows of $\mathbf{X}$. In a different context, the idea to measure the distance between configurations through their interpoint distances has been studied extensively by Meulman (1986). We use (3.5) as the denominator of the tunneling function, which leads to the following tunneling function

$$\tau_3(\mathbf{X}) = \frac{\left(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)\right)^2}{\left\|\mathbf{D}(\mathbf{X}) - \mathbf{D}(\mathbf{X}^*)\right\|_{\mathbf{w}}^2}. \tag{3.6}$$

Another advantage of $\tau_3(\mathbf{X})$ is that both numerator and denominator are comparable entities. Both STRESS and the current denominator are (squared) norms of the difference between two vectors. By using (3.6) we have obtained invariance of the tunneling function under any rotation $\mathbf{R}$ of the configuration, which implies that the effectiveness of the pole is extended to all $\mathbf{X}$ in the neighbourhood of $\mathbf{X}^*\mathbf{R}$.

3.1.2 The pole strength

In this section we discuss the strength of the pole analytically and give a suggestion how to increase the pole strength. Groenen (1990) showed empirically that the pole (denominator of the tunneling function) is not strong enough to cancel out a zero point of the tunneling function at $\mathbf{X}^*$. In this section, we give an upper bound for $\tau_3(\mathbf{X})$. Furthermore, we shall prove analytically that the tunneling function $\tau_3(\mathbf{X})$ may tend to zero when $\mathbf{X}$ approaches $\mathbf{X}^*$.

The following theorem states that the numerator of $\tau_3(\mathbf{X})$ is always smaller than its denominator.

Theorem 3.1
For each $\mathbf{X}, \mathbf{X}^* \in \mathbf{R}^{np}$ it holds that $(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*))^2 \leq \|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_w^2$.

Proof
Let $\boldsymbol{\delta}$, $\mathbf{d}$, and $\mathbf{d}^*$ be the vectors of length $n(n-1)/2$ with lower triangular elements of $\boldsymbol{\Delta}$, $\mathbf{D}(\mathbf{X})$, and $\mathbf{D}(\mathbf{X}^*)$ respectively, so that $\sigma(\mathbf{X}) = \|\boldsymbol{\delta} - \mathbf{d}\|_w$ and $\|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_w^2 = \|\mathbf{d}^* - \mathbf{d}\|_w^2$. Applying the triangle inequality twice we obtain

$$\begin{aligned} \|\boldsymbol{\delta} - \mathbf{d}\|_w &\leq \|\boldsymbol{\delta} - \mathbf{d}^*\|_w + \|\mathbf{d} - \mathbf{d}^*\|_w, \\ \|\boldsymbol{\delta} - \mathbf{d}^*\|_w &\leq \|\boldsymbol{\delta} - \mathbf{d}\|_w + \|\mathbf{d} - \mathbf{d}^*\|_w. \end{aligned}$$

So $\|\mathbf{d} - \mathbf{d}^*\|_w$ must be larger than or equal to both $\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)$ and $\sigma(\mathbf{X}^*) - \sigma(\mathbf{X})$, and therefore larger than or equal to their maximum too:

$$|\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*)| \leq \|\mathbf{d} - \mathbf{d}^*\|_w.$$

Squaring both sides gives the desired result. Q.E.D.

The preceding inequality implies that $0 \leq \tau_3(\mathbf{X}) \leq 1$. This upper bound of $\tau_3(\mathbf{X})$ does not tell us whether the pole is strong enough to avoid a minimum at $\mathbf{X}^*$. It merely shows that the maximum value of the tunneling function equals one. Note that theorem 3.1 uses a part of the proof of theorem 2.1, which stated that STRESS has a Lipschitz constant.

Another interesting question is where the extreme values of $\tau_3(\mathbf{X})$ can be found. To answer this question we look at the behaviour of $\tau_3(\mathbf{X})$ as $\mathbf{X}$ approaches $\mathbf{X}^*$. Consider the complete space of vectors $\mathbf{d}$ in $\mathbf{R}^{n(n-1)/2}$. Although the set of vectors of Euclidean distances is a subset of $\mathbf{R}^{n(n-1)/2}$, we can investigate what happens when $\mathbf{d}$ approaches $\mathbf{d}^*$ from any direction $\mathbf{z}$. If $\mathbf{z}$ has unit norm, i.e., $\|\mathbf{z}\|_w = 1$, this amounts to developing the limit of $\tau_3(\mathbf{d}^* - r\mathbf{z})$ as r tends to 0;

$$\lim_{r\to 0} \tau_3(\mathbf{d}^*-r\mathbf{z}) = \lim_{r\to 0} \frac{\left(\|\boldsymbol{\delta}-\mathbf{d}^*+r\mathbf{z}\|_{\mathbf{w}} - \|\boldsymbol{\delta}-\mathbf{d}^*\|_{\mathbf{w}}\right)^2}{\|\mathbf{d}^*-\mathbf{d}^*+r\mathbf{z}\|_{\mathbf{w}}^2}$$

$$= \left(\lim_{r\downarrow 0} \frac{\|\boldsymbol{\delta}-\mathbf{d}^*+r\mathbf{z}\|_{\mathbf{w}} - \|\boldsymbol{\delta}-\mathbf{d}^*\|_{\mathbf{w}}}{r}\right)^2. \qquad (3.7)$$

By substituting $\mathbf{a} = \boldsymbol{\delta}-\mathbf{d}^*$, it can be seen that the latter equation is exactly the square of the directional derivative of the function $f(\mathbf{a}) = \|\mathbf{a}\|_{\mathbf{w}}$. If $\mathbf{g}$ is the gradient of $f(\mathbf{a})$, the directional derivative of $f(\mathbf{a})$ along direction $\mathbf{z}$ can be expressed as $\mathbf{g}'\mathbf{z}$ (see for example Gill, Murray, and Wright, 1981, p. 53). The gradient $\mathbf{g}$ of $f(\mathbf{a})$ is given by $\mathbf{D}_{\mathbf{w}}\mathbf{a}/\|\mathbf{a}\|_{\mathbf{w}}$, and the directional derivative is thus $\mathbf{a}'\mathbf{D}_{\mathbf{w}}\mathbf{z}/\|\mathbf{a}\|_{\mathbf{w}}$, where $\mathbf{D}_{\mathbf{w}}$ is a diagonal matrix with diagonal elements $\mathbf{w}$. Now we can write

$$\lim_{r\to 0} \tau_3(\mathbf{d}^*-r\mathbf{z}) = \left(\frac{(\boldsymbol{\delta}-\mathbf{d}^*)'\mathbf{D}_{\mathbf{w}}\mathbf{z}}{\|\boldsymbol{\delta}-\mathbf{d}^*\|_{\mathbf{w}}}\right)^2, \qquad (3.8)$$

which is between zero, when $\mathbf{z}$ is $\mathbf{D}_{\mathbf{w}}$-orthogonal to $\boldsymbol{\delta}-\mathbf{d}^*$, and one, when $\mathbf{z}$ is a multiple of $\boldsymbol{\delta}-\mathbf{d}^*$. In this proof we assumed that any $\mathbf{z}$ could be used from $\mathbb{R}^{n(n-1)/2}$. Clearly, the set of $\mathbf{d}^*-r\mathbf{z}$ that are Euclidean distance vectors is a subset of $\mathbb{R}^{n(n-1)/2}$. Since the proof holds for the whole set it must also hold for a subset. Therefore, we conclude that the value of $\tau_3(\mathbf{X})$ may be any value between zero and one as $\mathbf{X}$ approaches $\mathbf{X}^*$, though stronger results may exist for the subset.

Above, it was proven that $\tau_3(\mathbf{X})$ has values between 0 and 1 and that, unfortunately, near the pole the function also may have values between zero and one, depending from which direction we approach. Therefore, we cannot guarantee that the pole is strong enough to avoid a zero point of the tunneling function at the local minimum configuration of the STRESS function. One obvious way of creating a stronger pole is raising the denominator to some power larger than one, as suggested by Levy and Gomez (1985). Or, equivalently, we can take a root of the numerator using the pole strength parameter λ. Clearly, both options are possible, because they are monotonic transformations of each other, and such a transformation does not influence the minima and therefore the zero points of the tunneling function.

Having eliminated the rotational problem and the pole strength problem, we redefine the tunneling function by

$$\tau_4(\mathbf{X}) = \frac{(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*))^{2\lambda}}{\|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_{\mathbf{w}}^2}. \qquad (3.9)$$

In section 3.2 we shall see that taking the numerator to the power λ, with $0 < \lambda < 1$, is indeed a good choice for the majorization procedure for the tunneling function.

3.1.3 Attraction to the horizon

Here we show that it is of importance to see how the tunneling function behaves when the configuration moves to the horizon, i.e., when the coordinates get uniformly large. It turns out that during minimization of $\tau_4(\mathbf{X})$ the configuration may be attracted to the horizon because the tunneling function values tend to zero.

Let us first define what is understood by a large configuration. Given a configuration $\mathbf{X}$, not equal to $\mathbf{0}$, we say that $c\mathbf{X}$ is large when the uniform dilation factor c is large. The behaviour of the tunneling function for large configurations can now be reformulated as the mathematical problem of finding the limit of $\tau_4(c\mathbf{X})$ as c goes to infinity. For solving this limit we use the fact that the Euclidean distance function is homogenous, so that $d_{ij}(c\mathbf{X}) = c\,d_{ij}(\mathbf{X})$. Then the limit can be written as

$$\lim_{c\to\infty} \tau_4(c\mathbf{X}) = \lim_{c\to\infty} \frac{(\|\boldsymbol{\delta} - c\mathbf{d}\|_w - \|\boldsymbol{\delta} - \mathbf{d}^*\|_w)^{2\lambda}}{\|\mathbf{d}^* - c\mathbf{d}\|_w^2}. \tag{3.10}$$

Further, to evaluate the limit it is convenient to substitute $c = \alpha/(1-\alpha)$ and subsequently let α tend to 1. This simplifies the limit into

$$\begin{aligned}
\lim_{c\to\infty} \tau_4(c\mathbf{X}) &= \lim_{\alpha\to 1} \left(\frac{\|\boldsymbol{\delta} - \alpha/(1-\alpha)\mathbf{d}\|_w - \|\boldsymbol{\delta} - \mathbf{d}^*\|_w}{\|\mathbf{d}^* - \alpha/(1-\alpha)\mathbf{d}\|_w^{1/\lambda}} \right)^{2\lambda} \\
&= \lim_{\alpha\to 1} \left(\frac{(1-\alpha)^{-1}\|(1-\alpha)\boldsymbol{\delta} - \alpha\mathbf{d}\|_w - \|\boldsymbol{\delta} - \mathbf{d}^*\|_w}{(1-\alpha)^{-1/\lambda}\|(1-\alpha)\mathbf{d}^* - \alpha\mathbf{d}\|_w^{1/\lambda}} \right)^{2\lambda} \\
&= \lim_{\alpha\to 1} \left(\frac{(1-\alpha)^{1/\lambda-1}\|(1-\alpha)\boldsymbol{\delta} - \alpha\mathbf{d}\|_w}{\|(1-\alpha)\mathbf{d}^* - \alpha\mathbf{d}\|_w^{1/\lambda}} - \frac{(1-\alpha)^{1/\lambda}\|\boldsymbol{\delta} - \mathbf{d}^*\|_w}{\|(1-\alpha)\mathbf{d}^* - \alpha\mathbf{d}\|_w^{1/\lambda}} \right)^{2\lambda} \\
&= \lim_{\alpha\to 1} \left(\frac{(1-\alpha)^{1/\lambda-1}\|\mathbf{d}\|_w}{\|\mathbf{d}\|_w^{1/\lambda}} - \frac{(1-\alpha)^{1/\lambda}\|\boldsymbol{\delta} - \mathbf{d}^*\|_w}{\|\mathbf{d}\|_w^{1/\lambda}} \right)^{2\lambda} \\
&= \lim_{\alpha\to 1} \left((1-\alpha)^{1/\lambda-1}\|\mathbf{d}\|_w^{1-1/\lambda} - \frac{(1-\alpha)^{1/\lambda}\|\boldsymbol{\delta} - \mathbf{d}^*\|_w}{\|\mathbf{d}\|_w^{1/\lambda}} \right)^{2\lambda} \\
&= (0-0)^{2\lambda} = 0 .
\end{aligned} \tag{3.11}$$

The final part of the limit follows from the assumption that $0 < \lambda < 1$, so that $1/\lambda - 1$ is greater than zero, $1/\lambda$ is greater than one, and consequently $(1-\alpha)^{1/\lambda-1}$ and $(1-\alpha)^{1/\lambda}$ tend to zero for α approaching one. This derivation clearly shows that for large configurations $\tau_4(\mathbf{X})$ tends to zero. This feature is unattractive, because we transformed the original tunneling problem into the problem of finding a zero point of the tunneling function. The limit above shows that we can expect $\tau_4(\mathbf{X})$ to be close to zero not only near a

configuration with equal STRESS value, but also for large configurations. This may direct our algorithm inadvertently to the horizon.

The problem of attraction to the horizon indicates that a minimization algorithm based on (3.9) may fail. To remedy this problem we are looking for a pole that has high values if $\mathbf{X}$ is close to $\mathbf{X}^*$, also has high or constant nonzero values for large configurations, but retains the zero points at configurations with STRESS equal to the local minimum STRESS value. Let the numerator of the tunneling function (3.9) be presented by $N(\mathbf{X})$ and the denominator $\|\mathbf{D}(\mathbf{X}^*) - \mathbf{D}(\mathbf{X})\|_w^2$ by $P(\mathbf{X})$, so the function can be written as $N(\mathbf{X})/P(\mathbf{X})$. Instead of the factor $1/P(\mathbf{X})$, a factor is sought that is close to unity for large $\mathbf{X}$, and yields a large value for $\mathbf{X}$ close to $\mathbf{X}^*$. One such factor that solves the problem of attraction to the horizon is $(1 + 1/P(\mathbf{X}))$. Using this factor results in the tunneling function $\tau_1(\mathbf{X})$ as presented in section 3.1. In the sequel we shall refer to $\tau_1(\mathbf{X})$ whenever we use tunneling functions with one pole. In the next section an algorithm is derived that searches for the zero points of $\tau_1(\mathbf{X})$.

3.2 Minimizing the tunneling function

In the previous section a tunneling function was developed that has desirable properties. This section discusses a zero finding algorithm of this function. Since the zero points are also the global minima of the tunneling function, we can use a minimization algorithm to find a zero point. The minimization method used here is a combination of parametric programming and iterative majorization . First, we show that parametric programming remains valid if iterative majorization is applied. Then we discuss how a product of two functions can be majorized. Next a majorization inequality is presented for a root of x. These results are then applied to the tunneling function and an algorithm is given.

3.2.1 Convergence proof of majorized parametric programming

The tunneling function $\tau_1(\mathbf{X})$ can be considered as a ratio of two functions of $\mathbf{X}$ and hence minimization of $\tau_1(\mathbf{X})$ can be seen as a fractional programming problem. An algorithm to minimize such a function was proposed by Dinkelbach (1967), and used by Heiser (1981) and Groenen (1990). However, Dinkelbach's algorithm assumes that at each iteration the absolute minimum over $\mathbf{X}$ of an auxiliary function $F(q,\mathbf{X})$ can be obtained. Such an assumption is too strong for our purpose, and in this section we develop a more flexible version of the algorithm. More specifically, we prove that the algorithm is still convergent if we find the minimum of a function that majorizes $F(q,\mathbf{X})$.

The tunneling function $\tau_1(\mathbf{X})$ is given by $N(\mathbf{X})(1+P(\mathbf{X}))/P(\mathbf{X})$, or, for ease of notation, $\tau_1(\mathbf{X})$ equals $M(\mathbf{X})/P(\mathbf{X})$ where $M(\mathbf{X}) = N(\mathbf{X})(1+P(\mathbf{X}))$. We assume that $P(\mathbf{X}) > 0$. Then Dinkelbach's algorithm amounts to

1. initialize $q \leftarrow \tau_1(\mathbf{X}^0)$
2. $\mathbf{X}^+ \leftarrow$ argmin $F(q,\mathbf{X})$
3. $q^+ \leftarrow \tau_1(\mathbf{X}^+)$
4. If $q^+ < \omega$ then *stop* (with ω a pre-set small positive constant)
5. else set $q \leftarrow q^+$ and go to 2

where the auxiliary function $F(q,\mathbf{X})$ is defined by

$$F(q,\mathbf{X}) = M(\mathbf{X}) - q\,P(\mathbf{X}). \tag{3.12}$$

If we cannot find the argument that minimizes $F(q,\mathbf{X})$ in step 2 analytically, we need a more relaxed procedure that still guarantees convergence. Suppose we have a function $\hat{M}(\mathbf{X},\mathbf{Y})$ that majorizes the numerator of $\tau_1(\mathbf{X})$, and another function $\hat{P}(\mathbf{X},\mathbf{Y})$ that *minorizes* the denominator. Thus

$$\begin{aligned} \hat{M}(\mathbf{X},\mathbf{Y}) &\geq M(\mathbf{X}) \qquad \text{with } \hat{M}(\mathbf{Y},\mathbf{Y}) = M(\mathbf{Y}), \\ \hat{P}(\mathbf{X},\mathbf{Y}) &\leq P(\mathbf{X}) \qquad \text{with } \hat{P}(\mathbf{Y},\mathbf{Y}) = P(\mathbf{Y}). \end{aligned}$$

Then the parametric function

$$\hat{F}(q,\mathbf{X},\mathbf{Y}) = \hat{M}(\mathbf{X},\mathbf{Y}) - q\,\hat{P}(\mathbf{X},\mathbf{Y}) \tag{3.13}$$

majorizes $F(q,\mathbf{X})$ for all $q \geq 0$. We can now state the following theorem, which guarantees that the majorization version of the parametric programming algorithm always improves $\tau_1(\mathbf{X})$, unless $\mathbf{X}$ satisfies the necessary conditions for a stationary point.

Theorem 3.2:
Let $\mathbf{X}^+$ be a unique minimum of $\hat{F}(q,\mathbf{X},\mathbf{Y})$ for $q = \tau_1(\mathbf{Y})$. Then, either $\tau_1(\mathbf{X}^+) < \tau_1(\mathbf{Y})$, or $\tau_1(\mathbf{X}^+) = \tau_1(\mathbf{Y})$. In the latter case, $\mathbf{X}^+$ is a stationary point if the gradient of $F(\mathbf{X})$ exists at $\mathbf{X}^+$.

Proof
From the definition of $\mathbf{X}^+$ and from the properties of a majorizing function we have

$$F(q,\mathbf{X}^+) \leq \hat{F}(q,\mathbf{X}^+,\mathbf{Y}) < \hat{F}(q,\mathbf{Y},\mathbf{Y}) = F(q,\mathbf{Y})$$

for all $\mathbf{X}^+ \neq \mathbf{Y}$. Now the particular choice $q = \tau_1(\mathbf{Y})$ yields

$$F(\tau_1(\mathbf{Y}),\mathbf{Y}) = M(\mathbf{Y}) - \frac{M(\mathbf{Y})}{P(\mathbf{Y})}\,P(\mathbf{Y}) = 0,$$

and therefore the inequality becomes

$$M(\mathbf{X}^+) - \tau_1(\mathbf{Y})P(\mathbf{X}^+) < 0.$$

Using the equality $\tau_1(\mathbf{X}^+) = M(\mathbf{X}^+)/P(\mathbf{X}^+)$, we obtain

$$[\tau_1(\mathbf{X}^+) - \tau_1(\mathbf{Y})]\, P(\mathbf{X}^+) < 0.$$

Since $P(\mathbf{X}^+)$ is always nonnegative by assumption, we must have $\tau_1(\mathbf{X}^+) < \tau_1(\mathbf{Y})$. Now suppose $\tau_1(\mathbf{X}^+) = \tau_1(\mathbf{Y})$; this can only happen if

$$\hat{F}(\tau_1(\mathbf{Y}),\mathbf{X}^+,\mathbf{Y}) = \hat{F}(\tau_1(\mathbf{Y}),\mathbf{Y},\mathbf{Y}).$$

Since the majorizing function has a unique minimum, it must be true that $\mathbf{X}^+ = \mathbf{Y}$, and $\mathbf{X}^+$ should satisfy the stationary equation for the majorizing function. Setting the partial derivatives with respect to $\mathbf{X}$ of $\hat{F}(\tau_1(\mathbf{Y}),\mathbf{X},\mathbf{X})$ equal to zero, we get

$$\frac{\partial M(\mathbf{X})}{\partial \mathbf{X}} = \frac{M(\mathbf{Y})}{P(\mathbf{Y})}\frac{\partial P(\mathbf{X})}{\partial \mathbf{X}}\,.$$

For a minimum of $\tau_1(\mathbf{X})$ the stationary equation is

$$\frac{\partial M(\mathbf{X})}{\partial \mathbf{X}} P(\mathbf{X}) - \frac{\partial P(\mathbf{X})}{\partial \mathbf{X}} M(\mathbf{X}) = \mathbf{0}\,.$$

When $\mathbf{X}^+ = \mathbf{Y}$ these two conditions are equivalent. Q.E.D.

This result implies that step 2 of Dinkelbach's algorithm can be relaxed into finding the minimum of the majorization function, i.e., $\mathbf{X}^+ \leftarrow \operatorname{argmin} \hat{F}(q,\mathbf{X},\mathbf{Y})$.

The relaxed version of Dinkelbach's algorithm keeps convergence to a local minimum of our ratio function. However, some other nice properties of the original algorithm, like concavity of $F(q,\mathbf{X}^+)$ in q, can no longer be proven. The stopping criterion (step 4) must be changed, because it cannot be guaranteed that the algorithm finds a zero point of the tunneling function; it may get stuck in a higher local minimum of $\tau_1(\mathbf{X})$. Note that the theorem can be restated for maximizing a ratio function by reversing the signs of all inequalities.

3.2.2 *Majorizing the product of two functions*

One term of the auxiliary function $F(q,\mathbf{X})$, introduced by parametric programming, is a product of two functions. Here we show how such a product can be majorized by quadratic majorization.

Let the product $N(\mathbf{X})(1 + P(\mathbf{X}))$ be represented here by x_1x_2. The inequality

$$\left(\frac{x_1}{y_1} - \frac{x_2}{y_2}\right)^2 \geq 0 \qquad (3.14)$$

is always true for $y_1, y_2 > 0$, because any square of a real argument is nonnegative. Rewriting this inequality gives

$$\left(\frac{x_1}{y_1}\right)^2 + \left(\frac{x_2}{y_2}\right)^2 - 2\frac{x_1 x_2}{y_1 y_2} \geq 0 \quad \text{or}$$

$$x_1 x_2 \leq \frac{1}{2}\frac{y_2}{y_1} x_1^2 + \frac{1}{2}\frac{y_1}{y_2} x_2^2 \ . \tag{3.15}$$

Furthermore, the inequality becomes a strict equality when x_1 equals y_1 and x_2 equals y_2. However, this happens also whenever x_1 equals $x_2 y_1/y_2$, which can be derived from setting the inequality to zero or, equivalently, setting $x_1/y_1 = x_2/y_2$. Note that this does not complicate the majorization procedure. The only requirement for majorization is that the auxiliary function touches the original function at a supporting point (as 3.2 does) and is larger than or equal to the original function at other points. Note that (3.15) is a special case of the geometric-arithmetic mean inequality.

3.2.3 Majorizing a root of a positive value

Above, it was pointed out that we must raise the numerator of the tunneling function to the power λ, where $0 < \lambda < 1$. It was needed to avoid a zero point of the tunneling function at the previous local minimum. As said before, we wish to stay within the majorization framework that guarantees a convergent algorithm. Therefore, a majorization inequality is required for this root.

Observe that the function $f(x) = x^\lambda$ is concave for $x \geq 0$ and $0 < \lambda < 1$ due to its negative second derivative. This implies that any line touching the curve must have a tangent that is equal to the first derivative $f'(x) = \lambda x^{\lambda-1}$. Let us consider the line touching the curve at $x = 1$. Then we know that $x^\lambda \leq ax + b$, that the tangent equals $f'(1) = a = \lambda$ and that $f(1) = 1 = a + b$, so that $b = 1 - \lambda$. Thus we have the following inequality

$$x^\lambda \leq (1 - \lambda) + \lambda x \ , \tag{3.16}$$

which was derived in a different way by Hardy, Littlewood, and Pólya (1952). Replacing x by x/y yields the majorization equation

$$x^\lambda \leq (1 - \lambda)\, y^\lambda + \lambda y^{\lambda-1}\, x \ , \tag{3.17}$$

which is an example of linear majorization. The inequality (3.17) can be applied directly to majorize the numerator of the tunneling function, which is done in the next section.

3.2.4 The tunneling algorithm

The minimization method for the tunneling function (3.2) consists of two concepts. The first one, parametric programming, is used to minimize a ratio of two functions by means of a auxiliary function $F(q,\mathbf{X})$. In section 3.2.1 it was shown that convergence is retained when an $\mathbf{X}^+$ is found that yields a lower value of the auxiliary function. The concept of majorization is used to find such $\mathbf{X}^+$. The numerator of the tunneling function is a product of two functions, that can be majorized by the sum of the squares of each function (see section 3.2.2). Therefore, we change to minimizing the square root of the tunneling function, so that after majorization we have a sum of the two functions. Note that taking the square root (or any other monotonic transformation) does not change the position of the minimum $\tau_1(\mathbf{X})$ or, in other words, minimizing $\tau_1(\mathbf{X})$ is equivalent to minimizing $\sqrt{\tau_1(\mathbf{X})}$. In the remaining part of this section we change to

$$\sqrt{\tau_1(\mathbf{X})} = \frac{\sqrt{N(\mathbf{X})}\sqrt{1+P(\mathbf{X})}}{\sqrt{P(\mathbf{X})}} = \left|\sigma(\mathbf{X})-\sigma(\mathbf{X}^*)\right|^{\lambda} \frac{\sqrt{\left\|\mathbf{d}-\mathbf{d}^*\right\|_w^2+1}}{\left\|\mathbf{d}-\mathbf{d}^*\right\|_w}, \tag{3.18}$$

which results in the auxiliary function

$$F(q,\mathbf{X}) = \sqrt{N(\mathbf{X})}\sqrt{1+P(\mathbf{X})} - q\sqrt{P(\mathbf{X})}\ , \tag{3.19}$$

where q equals $\sqrt{\tau_1(\mathbf{Y})}$ if $\mathbf{Y}$ is the previous configuration from the iterative process. In the next sections, the various steps are taken that are needed to majorize (3.19).

3.2.4.1 Majorizing $\sqrt{N(\mathbf{X})}\sqrt{1+P(\mathbf{X})}$

The outer majorization of the product $\sqrt{N(\mathbf{X})}\sqrt{1+P(\mathbf{X})}$ is given by (3.15) and can be directly applied

$$\sqrt{N(\mathbf{X})}\sqrt{1+P(\mathbf{X})} \le \frac{1}{2}\sqrt{\frac{1+P(\mathbf{Y})}{N(\mathbf{Y})}}N(\mathbf{X}) + \frac{1}{2}\sqrt{\frac{N(\mathbf{Y})}{1+P(\mathbf{Y})}}(1+P(\mathbf{X}))\ . \tag{3.20}$$

Furthermore, the functions $N(\mathbf{X})$ and $P(\mathbf{X})$ need to be majorized themselves. Let us first focus on the majorization of $N(\mathbf{X})$, which can be expressed as $(\sigma(\mathbf{X})-\sigma(\mathbf{X}^*))^2$ raised to the power λ, with $0<\lambda<1$. From (3.17) the inequality

$$\begin{aligned}\left(\sigma(\mathbf{X})-\sigma(\mathbf{X}^*)\right)^{2\lambda} &\le (1-\lambda)\left(\sigma(\mathbf{Y})-\sigma(\mathbf{X}^*)\right)^{2\lambda} + \lambda\left(\sigma(\mathbf{Y})-\sigma(\mathbf{X}^*)\right)^{2(\lambda-1)}\left(\sigma(\mathbf{X})-\sigma(\mathbf{X}^*)\right)^2 \\ &\le (1-\lambda)N(\mathbf{Y}) + \lambda N(\mathbf{Y})^{1-1/\lambda}\left(\sigma(\mathbf{X})-\sigma(\mathbf{X}^*)\right)^2 \end{aligned} \tag{3.21}$$

is obtained. We are left with $(\sigma(\mathbf{X})-\sigma(\mathbf{X}^*))^2$ which can also be majorized. Let us write it as $\sigma^2(\mathbf{X})+\sigma^2(\mathbf{X}^*)-2\sigma(\mathbf{X})\sigma(\mathbf{X}^*)$. The part $-\sigma(\mathbf{X})=-\|\delta-\mathbf{d}\|_w$ is majorized by the Cauchy-Schwarz inequality

$$-\sigma(\mathbf{X})\leq-\frac{(\delta-\mathbf{d})'\mathbf{D}_w(\delta-\mathbf{d}_Y)}{\sigma(\mathbf{Y})} \tag{3.22}$$

where $\mathbf{D}_w$ is a diagonal matrix with elements given by the vector of weights w, and $\mathbf{d}_Y$ is the vector with elements $d_{ij}(\mathbf{Y})$. Using the majorization inequality (3.22) shows that $(\sigma(\mathbf{X})-\sigma(\mathbf{X}^*))^2$ is always smaller than (or at most equal to)

$$\begin{aligned}
&\sigma^2(\mathbf{X}^*)+\|\delta-\mathbf{d}\|_w^2-2\phi(\delta-\mathbf{d})'\mathbf{D}_w(\delta-\mathbf{d}_Y)=\\
&\sigma^2(\mathbf{X}^*)+\|\delta\|_w^2+\|\mathbf{d}\|_w^2-2\delta'\mathbf{D}_w\mathbf{d}-2\phi\left(\|\delta\|_w^2-\delta'\mathbf{D}_w\mathbf{d}_Y-\delta'\mathbf{D}_w\mathbf{d}+\mathbf{d}_Y\mathbf{D}_w\mathbf{d}\right)=\\
&\sigma^2(\mathbf{X}^*)+(1-2\phi)\|\delta\|_w^2+\|\mathbf{d}\|_w^2-2(1-\phi)\delta'\mathbf{D}_w\mathbf{d}+2\phi\delta'\mathbf{D}_w\mathbf{d}_Y-2\phi\mathbf{d}_Y\mathbf{D}_w\mathbf{d},
\end{aligned} \tag{3.23}$$

where ϕ denotes $\sigma(\mathbf{X}^*)/\sigma(\mathbf{Y})$ for convenience of notation. If $\sigma(\mathbf{Y})\geq\sigma(\mathbf{X}^*)$, then the factor $(1-\phi)$ is nonnegative which implies that the term $-2(1-\phi)\delta'\mathbf{D}_w\mathbf{d}$ remains concave and can be majorized using standard SMACOF theory (see section 1.3) by $-2(1-\phi)\mathrm{tr}\,\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y}$. The same holds for the term $-2\phi\,\mathbf{d}'\mathbf{D}_w\mathbf{d}_Y=-2\phi\sum_{i<j}w_{ij}d_{ij}(\mathbf{Y})d_{ij}(\mathbf{X})$, for which a nice simplification occurs. Straightforward use of the Cauchy-Schwarz inequality gives $-d_{ij}(\mathbf{Y})d_{ij}(\mathbf{X})\leq-(\mathbf{x}_i-\mathbf{x}_j)'(\mathbf{y}_i-\mathbf{y}_j)$. Thus, we have the majorization inequality

$$-2\phi\mathbf{d}'\mathbf{D}_w\mathbf{d}_Y\leq-2\phi\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{Y}. \tag{3.24}$$

Combining these results with (3.23) leads to the following majorization inequality

$$\left(\sigma(\mathbf{X})-\sigma(\mathbf{X}^*)\right)^2\leq c_1+\mathrm{tr}\,\mathbf{X}'\mathbf{V}\mathbf{X}-2(1-\phi)\mathrm{tr}\,\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y}-2\phi\,\mathrm{tr}\,\mathbf{X}'\mathbf{V}\mathbf{Y}, \tag{3.25}$$

where

$$c_1=\sigma^2(\mathbf{X}^*)+(1-2\phi)\|\delta\|_w^2+2\phi\delta'\mathbf{D}_w\mathbf{d}_Y. \tag{3.26}$$

To simplify the notation, we shall use constants c_i to indicate all terms that are not a function of $\mathbf{X}$. The restriction $\sigma(\mathbf{Y})\geq\sigma(\mathbf{X}^*)$ does not impose an extra problem, because where ever it is violated, the STRESS must be smaller than the STRESS of the previous local minimum, which is the main goal of the tunneling step in the first place.

We continue with the majorization of $\mathrm{N}(\mathbf{X})$. Inserting (3.25) in (3.21) yields

$$\begin{aligned}
\mathrm{N}(\mathbf{X})\leq\;&(1-\lambda)\mathrm{N}(\mathbf{Y})+\\
&\lambda\,\mathrm{N}(\mathbf{Y})^{1-1/\lambda}\left(c_1+\mathrm{tr}\,\mathbf{X}'\mathbf{V}\mathbf{X}-2(1-\phi)\mathrm{tr}\,\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y}-2\phi\,\mathrm{tr}\,\mathbf{X}'\mathbf{V}\mathbf{Y}\right).
\end{aligned} \tag{3.27}$$

The majorization of P(**X**) is analogous to the majorization of the STRESS function itself. We only give the result here and refer to section 1.3 for details. The function P(**X**) can be majorized by the inequality

$$P(\mathbf{X}) \leq \text{tr}\mathbf{X}^{*\prime}\mathbf{V}\mathbf{X}^{*} + \text{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\text{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}^{*}),\mathbf{Y})\mathbf{Y}\,, \tag{3.28}$$

where $\mathbf{B}(\mathbf{D}(\mathbf{X}^{*}),\mathbf{Y})$ has off diagonal elements $b_{ij} = -w_{ij}d_{ij}(\mathbf{X}^{*})/d_{ij}(\mathbf{Y})$ and diagonal elements $b_{ii} = -\sum_{i\neq j}b_{ij}$.

To conclude this section, the previous results are substituted in (3.20) so that $\sqrt{N(\mathbf{X})}\sqrt{1+P(\mathbf{X})}$ is majorized by a function that has only quadratic and linear terms in **X**. The substitution yields

$$\begin{aligned}\sqrt{N(\mathbf{X})}\sqrt{1+P(\mathbf{X})} \leq\ & \\ & \tfrac{1}{2}\lambda(1+P(\mathbf{Y}))^{1/2}N(\mathbf{Y})^{1/2-1/\lambda}(\text{tr}\,\mathbf{X}'\,\mathbf{V}\mathbf{X} - 2(1-\phi)\text{tr}\,\mathbf{X}'\,\mathbf{B}(\mathbf{Y})\mathbf{Y} - 2\phi\,\text{tr}\,\mathbf{X}'\,\mathbf{V}\mathbf{Y}) + \\ & \tfrac{1}{2}N(\mathbf{Y})^{1/2}(1+P(\mathbf{Y}))^{-1/2}\left(\text{tr}\,\mathbf{X}'\,\mathbf{V}\mathbf{X} - 2\,\text{tr}\,\mathbf{X}'\,\mathbf{B}(\mathbf{D}(\mathbf{X}^{*}),\mathbf{Y})\mathbf{Y}\right) + c_2\,.\end{aligned} \tag{3.29}$$

This inequality is used in section 3.2.4.3 for the majorization of F(q,**X**).

3.2.4.2 Majorizing $-q\sqrt{P(\mathbf{X})}$

The term $-q\sqrt{P(\mathbf{X})}$ can be expressed as $-q\|\mathbf{d}^{*} - \mathbf{d}\|$. Such a negative Euclidean norm can be majorized using the Cauchy-Schwarz inequality, i.e.,

$$(\mathbf{d}^{*} - \mathbf{d})'\mathbf{D}_w(\mathbf{d}^{*} - \mathbf{d}_\mathbf{Y}) \leq \|\mathbf{d}^{*} - \mathbf{d}\|_w\,\|\mathbf{d}^{*} - \mathbf{d}_\mathbf{Y}\|_w\,, \tag{3.30}$$

where $\mathbf{d}_\mathbf{Y}$ is a vector of distances between the points of configuration **Y**. The inequality becomes an equality if $\mathbf{d}_\mathbf{Y}$ equals **d**. Dividing both sides of (3.30) by $\|\mathbf{d}^{*} - \mathbf{d}_\mathbf{Y}\|_w$ and multiplying by –1 yields

$$\begin{aligned}-\|\mathbf{d}^{*} - \mathbf{d}\|_w \leq\ & -\frac{(\mathbf{d}^{*} - \mathbf{d})'\,\mathbf{D}_w(\mathbf{d}^{*} - \mathbf{d}_\mathbf{Y})}{\|\mathbf{d}^{*} - \mathbf{d}_\mathbf{Y}\|_w} = \frac{-\mathbf{d}^{*}\mathbf{D}_w\mathbf{d}^{*} + \mathbf{d}^{*\prime}\,\mathbf{D}_w\mathbf{d}_\mathbf{Y} + \mathbf{d}'\,\mathbf{D}_w\mathbf{d}^{*} - \mathbf{d}'\,\mathbf{D}_w\mathbf{d}_\mathbf{Y}}{\|\mathbf{d}^{*} - \mathbf{d}_\mathbf{Y}\|_w} \\ =\ & \frac{\mathbf{d}'\mathbf{D}_w\mathbf{d}^{*} - \mathbf{d}'\mathbf{D}_w\mathbf{d}_\mathbf{Y}}{\|\mathbf{d}^{*} - \mathbf{d}_\mathbf{Y}\|_w} + c_3\,.\end{aligned} \tag{3.31}$$

The term $\mathbf{d}'\mathbf{D}_w\mathbf{d}^{*}$, in sum notation $1/2\,\sum_{ij}w_{ij}d_{ij}(\mathbf{X}^{*})\,d_{ij}(\mathbf{X})$, can itself be majorized. More specifically, $d_{ij}(\mathbf{X})$ is majorized by $d_{ij}(\mathbf{Y})/2 + d_{ij}^2(\mathbf{X})/2d_{ij}(\mathbf{Y})$, using an inequality derived from (3.16) for $\lambda = 1/2$. Heiser (1991) notes that when $d_{ij}(\mathbf{Y})$ is zero, this majorizing function does not work. Therefore, he majorizes $d_{ij}(\mathbf{X})$ by $\varepsilon/2 + d_{ij}^2(\mathbf{X})/2\varepsilon$ whenever $d_{ij}(\mathbf{Y})$ is smaller than a small positive constant ε. This problem frequently arises when using quadratic majorization. In the sequel, we shall implicitly use this adaptation

whenever necessary. Multiplying $d_{ij}(\mathbf{X})$ and its majorizing function by $d_{ij}(\mathbf{X}^*)$, and summing over all ij, yields

$$\begin{aligned}\mathbf{d}'\mathbf{d}^* &\le \tfrac{1}{4}\textstyle\sum_{i<j} w_{ij} d_{ij}(\mathbf{X}^*)\, d_{ij}(\mathbf{Y}) + \tfrac{1}{4}\sum_{i<j} w_{ij}\frac{d_{ij}(\mathbf{X}^*)}{d_{ij}(\mathbf{Y})} d_{ij}^2(\mathbf{X}) \\ &= c_4 + \tfrac{1}{2}\,\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{X}\,. \end{aligned} \tag{3.32}$$

We have already seen in (3.24) that the term $-\mathbf{d}'\mathbf{D}_w\mathbf{d}_Y$ can be majorized by $-\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{Y}$. Combining the inequalities (3.31), (3.32) and (3.24) shows that the term $-q\sqrt{\mathrm{P}(\mathbf{X})}$ can be majorized as follows:

$$-q\sqrt{\mathrm{P}(\mathbf{X})} \le \tfrac{1}{2} q \frac{\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{Y}}{\|\mathbf{d}^* - \mathbf{d}_Y\|_w} + c_5\,. \tag{3.33}$$

The results from this section and the previous one are combined in the next section to present the update formula of the majorization algorithm for minimization of $\tau_1(\mathbf{X})$.

3.2.4.3 The update

From section 3.2.1 it is known that the tunneling function can be minimized iteratively by finding an update that yields a lower value of the auxiliary function $\mathrm{F}(q,\mathbf{X})$. The majorization inequalities are used to find such an update. Expressing $\mathrm{F}(q,\mathbf{X})$ as $\sqrt{\mathrm{N}(\mathbf{X})}\sqrt{1+\mathrm{P}(\mathbf{X})} - q\sqrt{\mathrm{P}(\mathbf{X})}$, where q equals $\sqrt{\tau_1(\mathbf{X})}$, let us combine the majorization results (3.29) and (3.33) into

$$\begin{aligned} e\,\mathrm{F}(q,\mathbf{X}) \le\ & \lambda[1+\mathrm{P}(\mathbf{Y})]\mathrm{N}(\mathbf{Y})^{1-1/\lambda}\,[\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2(1-\phi)\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y} - 2\phi n\,\mathrm{tr}\mathbf{X}'\mathbf{Y}] + \\ & + \mathrm{N}(\mathbf{Y})\,[\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{Y}] + \\ & + \tau_1(\mathbf{Y})\,[\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{Y}] + c_6, \end{aligned} \tag{3.34}$$

where e is the factor $2\sqrt{\mathrm{N}(\mathbf{Y})}\sqrt{1+\mathrm{P}(\mathbf{Y})}$ introduced here to simplify the notation. Further, let α equal $\lambda[1+\mathrm{P}(\mathbf{Y})]\mathrm{N}(\mathbf{Y})^{1-1/\lambda}$, β equal $\mathrm{N}(\mathbf{Y})$ and γ equal $\tau_1(\mathbf{Y})$. Then, the preceding inequality can be rewritten as

$$\begin{aligned} e\,\mathrm{F}(q,\mathbf{X}) \le\ & \alpha[\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2(1-\phi)\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y} - 2\phi\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{Y}] \\ & + \beta[\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{Y}] \\ & + \gamma[\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{Y}] + c_6 \\ =\ & \mathrm{tr}\mathbf{X}'[(\alpha+\beta)\mathbf{V} + \gamma\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})]\mathbf{X} \\ & - 2\mathrm{tr}\mathbf{X}'[\mathbf{B}(\alpha(1-\phi)\mathbf{\Delta} + \beta\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{Y} + (\alpha\phi+\gamma)\mathbf{V}\mathbf{Y}] + c_6 \\ =\ & \mathrm{tr}\mathbf{X}'\mathbf{M}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{Z} + c_6\,, \end{aligned} \tag{3.35}$$

where

$$\mathbf{M} = (\alpha+\beta)\mathbf{V} + \gamma\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y}), \text{ and} \tag{3.36}$$

$$\mathbf{Z} = \mathbf{B}(\alpha(1-\phi)\Delta + \beta\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{Y} + (\alpha\phi+\gamma)\mathbf{V}\mathbf{Y}. \tag{3.37}$$

Thus $F(q,\mathbf{X})$ is majorized by a quadratic function in $\mathbf{X}$. When $\mathbf{V}$ is positive semi-definite, the minimum of a quadratic function can be found in one step by setting the gradient to zero, i. e.

$$\nabla(\mathrm{tr}\mathbf{X}'\mathbf{M}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{Z} + c_6) = 2\mathbf{M}\mathbf{X} - 2\mathbf{Z} = \mathbf{0}, \tag{3.38}$$

which implies that $\mathbf{M}\mathbf{X} = \mathbf{Z}$ or $\mathbf{X} = \mathbf{M}^{-}\mathbf{Z}$. Note that $\mathbf{M}$ is of rank $n-1$, since $\mathbf{1}$ spans its null space and thus we use Moore-Penrose inverse $\mathbf{M}^{-} = (\mathbf{M} + \mathbf{1}\mathbf{1}')^{-1} - n^{-2}\mathbf{1}\mathbf{1}'$. Moreover, $\mathbf{M}$ is positive semi-definite, because it is the weighted sum of the positive semi-definite matrices $\mathbf{V}$ and $\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})$, where the weights $\alpha+\beta$ and γ are positive.

This leads us to the update formula needed in step 2 of the parametric programming algorithm: the update $\mathbf{X}^+$, given by

$$\mathbf{X}^+ = [(\alpha+\beta)\mathbf{V} + \gamma\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})]^{-}[\mathbf{B}(\alpha(1-\phi)\Delta+\beta\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{Y} + (\alpha\phi+\gamma)\mathbf{V}\mathbf{Y}], \tag{3.39}$$

ensures a lower value of $\tau_1(\mathbf{X})$ in every iteration until convergence is attained.

3.2.5 Acceleration

The previous section showed how to get an update. However, the algorithm seriously slows down whenever $\sigma(\mathbf{X})$ is close to $\sigma(\mathbf{X}^*)$. This is particularly a nuisance if we are close to a zero point of the tunneling function. Here we shall adapt the tunneling function once more to remedy this defect. The price of acceleration at the end of the tunneling step is that some of the theorems stated above do not necessarily hold anymore.

The slow down of the tunneling algorithm near a zero point is caused by the factors $(1-\phi)$ and ϕ in (3.39). For the moment we look at the situation that arises near the end of the tunneling step: $\sigma(\mathbf{X})$ is close to $\sigma(\mathbf{X}^*)$, $P(\mathbf{X})$ is sufficiently different from zero, and therefore $\tau_1(\mathbf{X})$ is also close to zero. Then, the update is dominated by the previous configuration $\mathbf{Y}$. This can be seen by examining the terms $\alpha(1-\phi)\mathbf{B}(\Delta,\mathbf{Y})\mathbf{Y}$ and $\alpha\phi\mathbf{V}\mathbf{Y}$ of (3.39) more closely. Since ϕ equals $\sigma(\mathbf{X}^*)/\sigma(\mathbf{Y})$, we have for $\sigma(\mathbf{Y})$ close to $\sigma(\mathbf{X}^*)$ that ϕ is close to one. Clearly, the term $\alpha(1-\phi)\mathbf{B}(\Delta,\mathbf{Y})\mathbf{Y}$ is dominated by the term $\alpha\phi\mathbf{V}\mathbf{Y}$. Further, it can be seen that α dominates the terms with β and γ, by examining the size and the ratio of α and β, and realizing that γ equals $\tau_1(\mathbf{Y})$, which is close to zero. Thus, the update changes slowly and convergence to the zero point of the tunneling function is unsatisfactorily slow.

The factor ϕ originates from majorizing $(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*))^2$. Clearly, using the function $|\sigma^2(\mathbf{X}) - \sigma^2(\mathbf{X}^*)|$ instead does not change the main characteristics of the tunneling function. It still has positive values everywhere and has zero points whenever $\sigma^2(\mathbf{X}) = \sigma^2(\mathbf{X}^*)$. Unfortunately, some of the other characteristics are lost. For example, we can not prove anymore that $|\sigma^2(\mathbf{X}) - \sigma^2(\mathbf{X}^*)| \leq P(\mathbf{X})$. However, the gain of acceleration supersedes this slight disadvantage. The adapted form of the tunneling function becomes

$$\tau_5(\mathbf{X}) = \left|\sigma^2(\mathbf{X}) - \sigma^2(\mathbf{X}^*)\right|^{\lambda} \left[1 + \frac{1}{\left\|\mathbf{D}(\mathbf{X}) - \mathbf{D}(\mathbf{X}^*)\right\|^2}\right]. \tag{3.40}$$

As an immediate consequence we have to change our minimization procedure. The first change concerns the majorization of $|\sigma^2(\mathbf{X}) - \sigma^2(\mathbf{X}^*)|$. Let $\tau_5(\mathbf{X})$ be defined by (3.40) for all $\mathbf{X}$ in set X_1 for which $\sigma^2(\mathbf{X}) \geq \sigma^2(\mathbf{X}^*)$ holds, and let $\tau_5(\mathbf{X}) = -\infty$ everywhere else, for $\mathbf{X}$ in the complementary set X_2. This allows us to remove the absolute signs in (3.40), since $\sigma^2(\mathbf{X}) - \sigma^2(\mathbf{X}^*) = |\sigma^2(\mathbf{X}) - \sigma^2(\mathbf{X}^*)|$ holds for the set X_1 by definition. Remember that we had the similar restriction $\sigma(\mathbf{Y}) \geq \sigma(\mathbf{X}^*)$ for the majorization inequality of $(\sigma(\mathbf{X}) - \sigma(\mathbf{X}^*))^2$, see (3.25). The accelerated tunneling function keeps all the characteristics discussed before for $\mathbf{X}$ in set X_1. Finding a point in set X_2 amounts to finding an $\mathbf{X}$ with $\sigma^2(\mathbf{X}) \leq \sigma^2(\mathbf{X}^*)$. This may be seen as an extension of the original tunneling step, where we searched for an $\mathbf{X}$ with $\sigma^2(\mathbf{X}) = \sigma^2(\mathbf{X}^*)$. Speaking in terms of the metaphor, when minimizing $\tau_5(\mathbf{X})$ we dig the tunnel horizontally or downwards. Note that all the majorization results remain true, as long as we stop whenever $\mathbf{X}$ is in set X_2. Using the previous results, an accelerated version of the update is

$$\mathbf{X}^+ = [(\alpha+\beta)\mathbf{V} + \gamma\mathbf{B}(\mathbf{D}(\mathbf{X}^*),\mathbf{Y})]^- [\mathbf{B}(\alpha\Delta+\beta\mathbf{D}(\mathbf{X}^*),\mathbf{Y})\mathbf{Y} + \gamma\mathbf{V}\mathbf{Y}]. \tag{3.41}$$

The acceleration happens if the tunneling function is close to a zero point. In the sequel, we shall use the accelerated version of the tunneling function $\tau_5(\mathbf{X})$ implicitly, whenever we refer to the one pole tunneling function.

3.3 A tunneling function with multiple poles

Finding the zero points of the tunneling function is done by minimizing $\tau_5(\mathbf{X})$. However, the majorization algorithm does not guarantee that the global minimum (the zero point) is found. Therefore, the algorithm may stop whenever the tunneling function has a stationary point with $\tau_5(\mathbf{X}) > 0$. To deal with this problem the tunneling function can be extended with an additional pole as to avoid such undesired stationary points of the tunneling function. Now we can proceed in two ways. First, we can follow Levy and Gomez (1985) who used a moving pole that is placed at a current stationary point of the

tunneling function not equal to zero. A second option is to introduce an additional pole at each new stationary point with $\tau_5(\mathbf{X}) > 0$. Once a stationary point is found, it is always avoided by the tunneling function. The surface of the tunneling function will have at least as many peaks as there are poles. We shall develop the latter option in this section.

In the case of one pole, discussed in section 3.2.4, we had to minimize $\sqrt{\tau_1(\mathbf{X})}$. For multiple poles we minimize

$$\sqrt{\tau_6(\mathbf{X})} = \sqrt{\mathrm{N}(\mathbf{X})}\,2^r\sqrt{\prod_{i=1}^{r}\left(1+\frac{1}{\mathrm{P}_i(\mathbf{X})}\right)} = \sqrt{\frac{\mathrm{N}(\mathbf{X})\prod_{i=1}^{r}\left(1+\mathrm{P}_i(\mathbf{X})\right)^{1/r}}{\prod_{i=1}^{r}\mathrm{P}_i(\mathbf{X})^{1/r}}}, \qquad (3.42)$$

where r is the number of poles. As shown in section 3.2.1 such a fraction can be minimized by majorizing the numerator of (3.42) minus q times the denominator. Thus, we have to find a generalization for the majorization of a product of functions and a generalization for minus the product of functions.

3.3.1 Majorization with multiple poles: the numerator

The numerator of the tunneling function with more than one pole can be regarded as the product of several functions. In this respect it is a generalization of the product of two functions. In section 3.2.2 a majorization inequality was found for the latter case. It turns out to be a special case of the inequality of the geometric mean and the ordinary mean. The same inequality is used here to majorize the product of several functions.

The inequality for the geometric mean and the arithmetic mean can be expressed as follows

$$\prod_{i=1}^{r} x_i^{1/r} \le \frac{1}{r}\sum_{i=1}^{r} x_i \quad , \qquad (3.43)$$

see, for example, Hardy et al. (1952). Note that this inequality assumes nonnegativity of all x_i since the root of a negative value is not defined in real space. For r equal to 2, (3.43) reduces to $\sqrt{x_1 x_2} \le (x_1 + x_2)/2$. Clearly, (3.43) is a strict equality when all x_i are unity or zero. However, the majorization theory requires equality when x equals y. This is achieved by replacing x_i by x_i/y_i. The obtained inequality can be multiplied on both sides by $\prod_{i=1}^{r} y_i^{1/r}$ which yields

$$\prod_{i=1}^{r} x_i^{1/r} \le \frac{1}{r}\prod_{i=1}^{r} y_i^{1/r}\sum_{i=1}^{r}\frac{x_i}{y_i} \quad . \qquad (3.44)$$

The majorization of a product of functions given by (3.44) can be applied directly to the numerator of the tunneling function with multiple poles. In its current form (3.44) is an example of linear majorization.

Majorizing the numerator of $\sqrt{\tau_6(\mathbf{X})}$ and using the majorization results of the mono pole case gives

$$\sqrt{N(\mathbf{X})}\sqrt{\prod_{i=1}^{r}(1+P_i(\mathbf{X}))^{1/r}} \le$$

$$\frac{1}{2}\lambda N(\mathbf{Y})^{1/2-1/\lambda}\ \prod_{i=1}^{r}[1+P_i(\mathbf{Y})]^{1/2r}\qquad [\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X}-2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y}]+$$

$$\frac{1}{2}N(\mathbf{Y})^{1/2}\qquad \prod_{i=1}^{r}[1+P_i(\mathbf{Y})]^{-1/2r}\frac{1}{r}\sum_{i=1}^{r}(1+P_i(\mathbf{Y}))^{-1}\ [\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X}-2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}_i^*),\mathbf{Y})\mathbf{Y}]$$

$$+c_7\ . \tag{3.45}$$

It is easy to see that for $r = 1$ this majorization results simplifies to the one derived for the one pole case.

3.3.2 Majorization with multiple poles: the denominator

Majorization of the denominator amounts to the majorization of minus a product of r functions. We can use the geometric-arithmetic mean inequality (3.43) again to find a majorization inequality that is quadratic in x_i.

In the previous section it was given that

$$0 \le \sqrt[r]{\prod_{i=1}^{r} x_i} \le \frac{1}{r}\sum_{i=1}^{r} x_i = \frac{1}{r}\mathbf{x}'\mathbf{1} \tag{3.46}$$

which is a strict equality if x equals **1**. The inequality that we shall use is

$$-\sqrt[r]{\prod_{i=1}^{r} x_i} \le \left(1-\frac{1}{r}\right)\mathbf{x}'\mathbf{x}-\left(2-\frac{1}{r}\right)\mathbf{x}'\mathbf{1}+(r-1), \tag{3.47}$$

which is an application of quadratic majorization. Thus we have to prove that

$$0 \le \sqrt[r]{\prod_{i=1}^{r} x_i}+\left(1-\frac{1}{r}\right)\mathbf{x}'\mathbf{x}-\left(2-\frac{1}{r}\right)\mathbf{x}'\mathbf{1}+(r-1) \tag{3.48}$$

holds. From the geometric-arithmetic mean inequality it must be true that

$$0 \le r\sqrt{\prod_{i=1}^{r} x_i} + \left(1-\frac{1}{r}\right)\mathbf{x}'\mathbf{x} - \left(2-\frac{1}{r}\right)\mathbf{x}'\mathbf{1} + (r-1)$$
$$\le \frac{1}{r}\mathbf{x}'\mathbf{1} + \left(1-\frac{1}{r}\right)\mathbf{x}'\mathbf{x} - \left(2-\frac{1}{r}\right)\mathbf{x}'\mathbf{1} + (r-1)\,, \qquad (3.49)$$

with a strict equality when r equals one, or $\mathbf{x}$ equals $\mathbf{1}$. The second part of (3.49) may be rewritten as

$$0 \le \left(1-\frac{1}{r}\right)\mathbf{x}'\mathbf{x} - 2\left(1-\frac{1}{r}\right)\mathbf{x}'\mathbf{1} + \left(1-\frac{1}{r}\right)\mathbf{1}'\mathbf{1}$$
$$0 \le \left(1-\frac{1}{r}\right)(\mathbf{x}-\mathbf{1})'(\mathbf{x}-\mathbf{1})\,, \qquad (3.50)$$

which is clearly always nonnegative, since $(\mathbf{x}-\mathbf{1})'(\mathbf{x}-\mathbf{1})$ is a sum of squares and $(1-\frac{1}{r})$ is positive for $r > 1$. This proves that (3.47) holds. Equality occurs for $\mathbf{x}$ equal to $\mathbf{1}$. The inequality can be easily transformed in a majorization inequality by applying the same idea as in section 3.3.1 (replacing x_i by x_i/y_i and multiplying both sides by $\prod_{i=1}^{n} y_i$). This gives

$$-r\sqrt{\prod_{i=1}^{r} x_i} \le r\sqrt{\prod_{i=1}^{r} y_i}\left(\left(1-\frac{1}{r}\right)\sum_{i=1}^{r}\frac{x_i^2}{y_i^2} - \left(2-\frac{1}{r}\right)\sum_{i=1}^{r}\frac{x_i}{y_i} + (r-1)\right)\,. \qquad (3.51)$$

The inequality derived above is used to majorize the $-q\sqrt{\prod_{i=1}^{r}\mathrm{P}_i(\mathbf{X})^{1/r}}$. Substituting directly into (3.51), we get

$$-q\sqrt{\prod_{i=1}^{r}\mathrm{P}_i(\mathbf{X})^{1/r}} \le$$
$$q\sqrt{\prod_{i=1}^{r}\mathrm{P}_i(\mathbf{Y})^{1/r}}\left((1-\frac{1}{r})\sum_{i=1}^{r}\frac{\mathrm{P}_i(\mathbf{X})}{\mathrm{P}_i(\mathbf{Y})} - (2-\frac{1}{r})\sum_{i=1}^{r}\sqrt{\frac{\mathrm{P}_i(\mathbf{X})}{\mathrm{P}_i(\mathbf{Y})}} + r - 1\right) \qquad (3.52)$$

From the majorization of $\tau_1(\mathbf{X})$ it is known that $\mathrm{P}_i(\mathbf{X})$ may be majorized by $\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}_i^*),\mathbf{Y})\mathbf{Y} + c$ and that $-\mathrm{P}_i^{1/2}(\mathbf{X})$ can be majorized by $1/2\ \mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}_i^*),\mathbf{Y})\mathbf{X} - 2\mathrm{P}_i(\mathbf{Y})^{-1/2}\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{Y} + c_5$. Thus

$$-q\sqrt{\prod_{i=1}^{r}\mathrm{P}_i(\mathbf{X})^{1/r}} \le$$
$$q\sqrt{\prod_{i=1}^{r}\mathrm{P}_i(\mathbf{Y})^{1/r}}\ (1-\frac{1}{r})\sum_{i=1}^{r}\mathrm{P}_i(\mathbf{Y})^{-1}\ [n\mathrm{tr}\mathbf{X}'\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}_i^*),\mathbf{Y})\mathbf{Y}] +$$
$$q\sqrt{\prod_{i=1}^{r}\mathrm{P}_i(\mathbf{Y})^{1/r}}\ \frac{1}{2}(2-\frac{1}{r})\sum_{i=1}^{r}\mathrm{P}_i(\mathbf{Y})^{-1}\ [\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}_i^*),\mathbf{Y})\mathbf{X} - 2n\mathrm{tr}\mathbf{X}'\mathbf{Y}] + c\,. \qquad (3.53)$$

Combining (3.53) and (3.45), multiplying both sides by $e = 2\mathrm{N}(\mathbf{Y})^{-1/2}\prod_{i=1}^{r}[1+\mathrm{P}_i(\mathbf{Y})]^{-1/2r}$ and by using $q = \tau_5^{1/2}(\mathbf{Y})$ we get

$$
\begin{aligned}
e\,F(q,\mathbf{X}) \;\leq\; & \\
& \lambda N(\mathbf{Y})^{-1/\lambda} && [\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y}] \\
+\; & \tfrac{1}{r}\sum_{i=1}^{r}(1+P_i(\mathbf{Y}))^{-1} && [\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}_i^*),\mathbf{Y})\mathbf{Y}] \\
+\; & 2(1-\tfrac{1}{r})\sum_{i=1}^{r} P_i^{-1}(\mathbf{Y}) && [\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}_i^*),\mathbf{Y})\mathbf{Y}] \\
+\; & (2-\tfrac{1}{r})\sum_{i=1}^{r} P_i^{-1}(\mathbf{Y}) && [\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{D}(\mathbf{X}_i^*),\mathbf{Y})\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{Y}] + c\,. \qquad (3.54)
\end{aligned}
$$

An explicit update formula can be obtained similarly as in section 3.2.4.3 by solving the system of equations obtained by setting the derivatives of the right hand part of (3.54) to zero. Note that for $r = 1$ this formula simply reduces to the one we have found for the one pole tunneling function.

3.4 Empirical results

After theoretical considerations in the previous sections, we present here the results of two empirical studies. The examples are meant to illustrate the tunneling algorithm.

The first small example, studied extensively by De Leeuw (1988), concerns a 4×4 dissimilarity matrix, with all dissimilarities equal to $1/\sqrt{6}$. He reports three stationary two-dimensional configurations $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$: $\mathbf{X}_1$ has four points on a line with $\sigma(\mathbf{X}_1) = 0.4082482905$, $\mathbf{X}_2$ has three points in the corners of an equilateral triangle and a point in the centroid with $\sigma(\mathbf{X}_2) = 0.2588190451$, and $\mathbf{X}_3$ has four points in the corners of a square with $\sigma(\mathbf{X}_3) = 0.1691019787$. (Note that De Leeuw reports the square of the STRESS value, i.e., $\sigma^2(\mathbf{X})$.)

We start the tunneling algorithm from $\mathbf{X}_1$ (see Figure 3.3a), which is a stationary point in the unidimensional scaling solution and a saddle point in two dimensions. The first objective for the tunneling algorithm is to find another configuration with STRESS 0.4082482905. The tunneling algorithm used a starting configuration that is a sum of the unidimensional scaling solution and a random matrix. The latter is needed because we have to start tunneling from a different point than $\mathbf{X}_1$ (since $\tau_6(\mathbf{X}_1)$ is undefined) and because we have to increase the rank of the solution from one to two. The pole strength parameter λ was set to $1/4$. After 85 iterations a solution was found with the same STRESS, as can be seen in Figure 3.2 that shows the history of iterations of the tunneling function values $\tau_5(\mathbf{X})$, the subsequent STRESS values $\sigma(\mathbf{X})$, and the values of the pole $P(\mathbf{X})$. After the first tunneling step, configuration b. of Figure 3.3 was found which has a STRESS of 0.4082483676. This solution differs in the seventh decimal position from the local minimum configuration $\mathbf{X}_1$. A local search from Figure 3.3b ended in the square configuration (see Figure 3.3c) which has STRESS 0.1691019835 which differs only in the eighth decimal place from the STRESS reported by De Leeuw. The small rise in the tunneling function value at the end occurs in the final iteration, because an $\mathbf{X}$ is

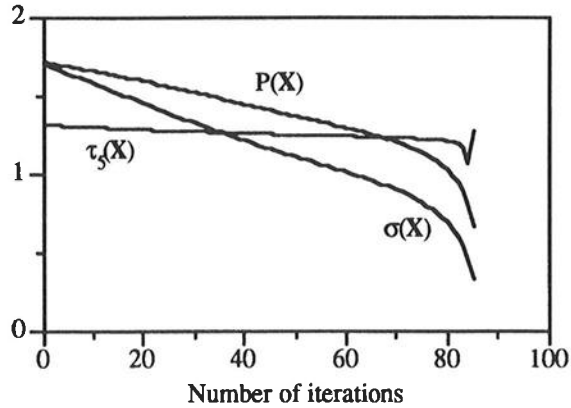
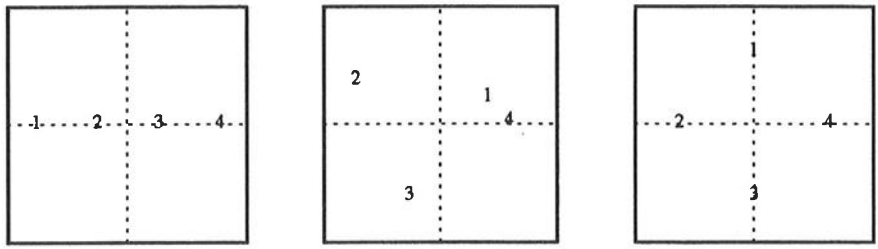


Figure 3.2 *The sequence of tunneling function values* $\tau_5(\mathbf{X})$, *STRESS values* $\sigma(\mathbf{X})$ *and value of the pole* $P(\mathbf{X})$ *against the iteration number.*



a b c

Figure 3.3 *Three configurations of the constant dissimilarity matrix: a. an unidimensional starting configuration, b. a configuration with the same STRESS as the unidimensional configuration, c. a lower local minimum of the STRESS function.*

found with STRESS lower than the previous local minimum. It has been pointed out in section 3.2.5 that the majorization algorithm only fails when $\sigma(\mathbf{X}) < \sigma(\mathbf{X}^*)$, which does not matter since the goal of the tunneling step has been reached.

The second stationary point, an equilateral triangle with centroid (the solution given by classical scaling), is a local minimum. Starting the tunneling algorithm here yields also the square configuration after one tunneling step and one local optimization step. This leads to two conclusions: a. the tunneling algorithm seems to work for this small example, and b. the square configuration attracts the algorithm strongly enough for both starting configurations. We return to this example in section 4.1.3.

The second example originates from Robinson (1951), and is also analyzed by Hubert and Arabie (1986); the data come from the Mani collection of archaeological

deposits. The dissimilarities are normalized to have sum of squares n^2. A stationary point was found with STRESS 0.88376265, and is shown in Figure 3.5a. Note that point 6, 7 and 8 are very close to each other. From a small perturbation of this point the tunneling algorithm was started. After 119 iterations the tunneling algorithm reached a configuration with STRESS 0.8557903127, which is indeed lower than the previous local minimum. Figure 3.4 displays a summary of the iterations. The small rise in the tunneling function value at the end happens here too.

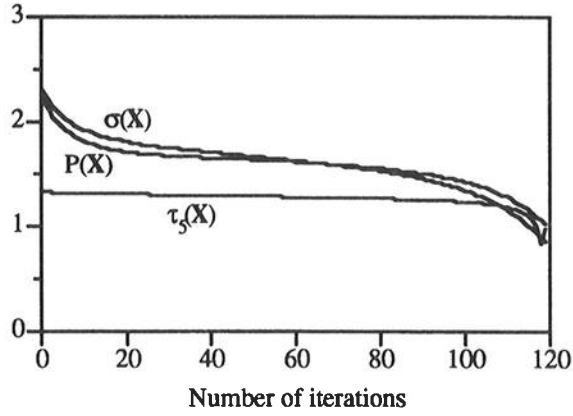


Figure 3.4 *The sequence of tunneling function values* $\tau_5(\mathbf{X})$, *STRESS values* $\sigma(\mathbf{X})$ *and value of the pole* $\mathbf{P}(\mathbf{X})$ *against the iteration number in the first tunneling step on data of the Mani collection.*

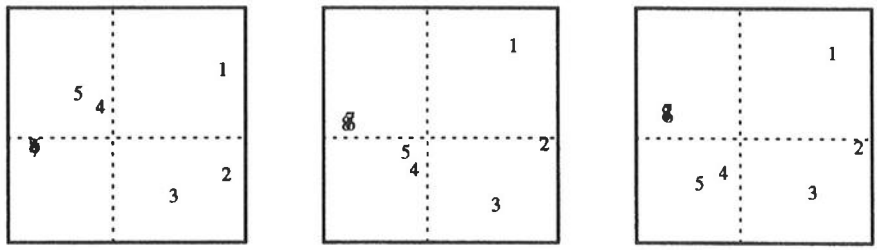


Figure 3.5 *Three configurations of the Mani collection data: a. stationary starting configuration, b. a configuration with the same STRESS as the starting configuration, c. a new local minimum with lower value of the STRESS function.*

After the tunneling step, one local search step was performed that yielded the configuration displayed in Figure 3.5c with STRESS 0.64722524. The tunneling algorithm was not able to find another configuration with STRESS 0.64722524. The three configurations in Figure 3.5 show two parts of the configuration: one group with points 1, 2 and 3 and the other group with points 4, 5, 6, 7 and 8. It seems that the second group needed to be reflected along the horizontal axis to obtain lower STRESS.

3.5 Fine tuning of the tunneling function

In the previous section we have seen that the tunneling method and its minimization algorithm works, at least for some small problems. In the examples, we used a pole strength parameter of 1/4. However, we do not know *a priori* which pole strength would be best. Another problem has to do with the normalization of $\boldsymbol{\delta}$, i.e., $\eta_\delta^2 = \boldsymbol{\delta}'\mathbf{D}_w\boldsymbol{\delta}$. When minimizing STRESS this normalization does not matter, because one Guttman transform automatically scales the configuration to the right size. This property does not hold anymore in the updates of the tunneling algorithm. To see how the tunneling function performs under various choices of pole strength and normalization, we do a small experiment where these factors are varied systematically. From this experiment, we can get a feeling for the usable values of these parameters.

The experiment has three factors: two different datasets of dissimilarities, various values of the pole strength parameter λ and different values of the normalization constant η_δ^2. The tunneling algorithm is started from a local minimum configuration that is known to be higher than the global minimum. To be more precise, for each combination of λ and η_δ^2 the same, properly scaled, start configuration is used, which is the sum of the local minimum and a fixed random perturbation. To rate the efficiency of the tunneling algorithm we compare the number of iterations needed to end the tunneling step. If it took more than 1000 iterations we stopped the tunneling step and regard it as having failed.

The first data set that is used was the Mani collection, also discussed in the previous section. The second data set consists of distances between 9 points in a regular two dimensional grid. We are focusing on the tunneling step that moves away from the local minimum configuration given in Figure 3.5a. The pole strength parameter was varied in 18 steps from 1/1.5, 1/2, 1/2.5, ..., 1/10. The normalization η_δ^2 varied in steps of 2 from 1 to 65 for the Mani data and 1 to 81 for the grid data. A three dimensional graph of the results of both data sets is shown in Figure 3.6. In the plot the number of iterations needed to stop the tunneling step are given against different values of $1/\lambda$ and the normalization of the dissimilarities η_δ^2.

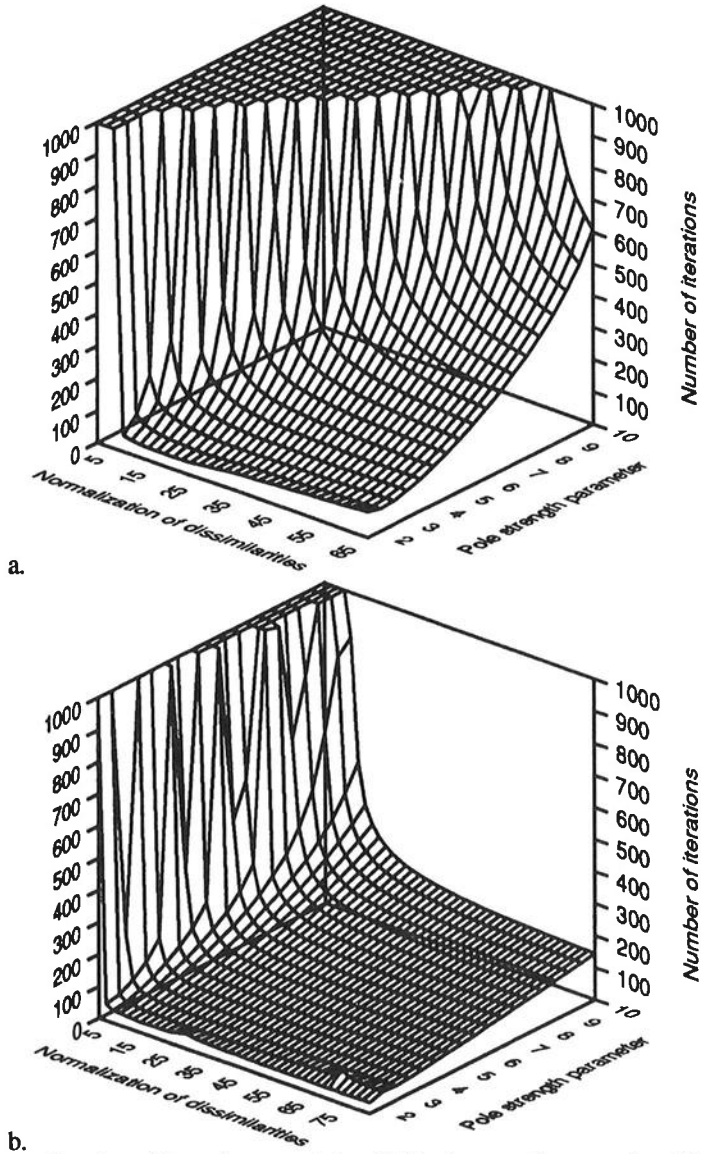


Figure 3.6 *Number of iterations needed to finish the tunneling step for different values of the pole strength parameter $1/\lambda$ and different normalizations of the dissimilarities η_δ^2. For every combination the tunneling step was truncated if 1000 iterations was exceeded. The dissimilarities in plot a. stem from the Mani collection, those in plot b. from a regular two dimensional grid of 9 points.*

CHAPTER 4

COMPARING SOME GLOBAL OPTIMIZATION METHODS

In this chapter we compare the performance of some of the global optimization methods that we have discussed in the previous chapters. Furthermore, we perform a number of simulation studies to find factors that may influence the occurrence of local minima. In particular, we study the effect of dimensionality, number of objects, and different error levels on the local minimum problem. We also look at the differences between the local minimum configurations themselves. The multidimensional case is treated separately in this chapter from the unidimensional case, since we found in chapter 1 that the unidimensional scaling can be reformulated as a combinatorial problem. We exclude full-dimensional scaling from our discussion in this chapter, since in section 1.4 full-dimensional scaling was expressed as a convex problem, which has only global minima. This chapter is not intended to give an extensive simulation study treating *all* possible factors influencing the local minimum problem. It merely tries to give some indication which factors could be of influence.

For multidimensional scaling, we compare tunneling with the multi-level-single-linkage (MLSL) clustering method. For unidimensional scaling, we compare four pairwise interchange strategies and the tabu search. A small example illustrates that the tunneling method still works for unidimensional scaling, albeit not perfectly.

4.1 Numerical experiments for multidimensional scaling

We start by investigating some of the factors that might influence the local minimum problem. The factors we discuss are the number of objects n, the dimensionality p, and the amount of error on the dissimilarities. Then we pick out some of the combinations for which many local minima occur, and apply the tunneling method. In the last part of this section we compare the tunneling method with the MLSL method on three specific datasets. Note that, as indicated before, the dimensionality is restricted to be $2 \leq p \leq n-1$ in this section.

4.1.1 *Investigating the local minimum problem for $p \geq 2$*

Here, we report a numerical experiment on the seriousness of the local minimum problem for dissimilarity matrices with a true underlying configuration for different dimensionality p, number of objects n, and perturbation by error. Our hypothesis is that as p gets larger (with respect to n) less local minima occur. This hypothesis is based on the fact that for $p = 1$ many local minima occur, and for $p = n - 1$ no local minima occur. We also expect that the number of local minima grows as the amount of error imposed on the dissimilarities increases.

The experiment was performed as follows. For each combination of the first two of the following three factors,

a. size of n, i.e., 10, 20, 40 and 100,
b. different values for the dimensionality p, i.e., 2, 3, 5 and 10,
c. different amount of error, i.e., 0%, 10%, 25%, and 100%,

a random configuration matrix of size $n \times p$ was generated. The resulting distance matrix was perturbed by error and used as a dissimilarity matrix, with the amount of error varying according to the levels of (c) above. We call such a dissimilarity matrix a gauge in the sequel. For each gauge, the SMACOF algorithm is started a hundred times from random $n \times p$ start configurations. We simply registered for each gauge which configuration yielded the lowest STRESS, and how often a local search ended in this candidate global minimum. In this way, we get an idea of the region of attraction of the candidate global minimum. Clearly, if no error is imposed the lowest local minimum has zero STRESS and hence is the global minimum.

It is not immediately clear how error should be added to the distance matrix to obtain a proper gauge. Several forms of error and error distributions could be used. For example, the dissimilarities could be a sum of true distances and error following some error distribution. If the error distribution has zero mean, as is the case with the normal distribution, this option has the disadvantage of possibly introducing negative dissimilarities. Although the algorithm could be adapted to deal with negative dissimilarities (see Heiser, 1991), we do not wish to introduce this latent factor in our simulation study. Negative dissimilarities are avoided by imposing error as Ramsay (1969) does, who assumes that the δ_{ij} is the Euclidean distance between a fixed part of the configuration coordinates and a part that changes for every pair. One objection to this model is that it produces dissimilarities which are always larger than the distances of the fixed coordinates. This problem is also avoided by using multiplicative error from a log-normal distribution as proposed by Wagenaar and Padmos (1971). True distances are generated and subsequently multiplied by errors of a log-normal distribution, which are positive so that no negative dissimilarities occur. In this study we have chosen the last error model to generate our gauges.

The construction of the gauges can be summarized as follows. Given the dimensionality p and the number of objects n, a configuration matrix is constructed with

We see that a small normalization and a large pole strength parameter increases the number of iterations needed to finish the tunneling step. Even an average pole strength value of 1/5 is not strong enough to finish the tunneling step in a configuration with lower STRESS in the Mani example. In both examples a small normalization of the dissimilarities performs badly, irrespective of the pole strength parameter. It seems that for stronger poles we also need to have larger normalization values. These examples suggest the use of a moderate pole strength parameter, like 1/3, and a normalization that is relatively large, for example $n(n-1)/2$. We shall use these values in the sequel.

3.6 Conclusions

In this chapter the tunneling method was presented, which aims at finding an ever decreasing series of local minima. The method iterates between a local search and a tunneling step in which a configuration is sought with equal or less STRESS than the previous local minimum. We changed the tunneling function used in the tunneling step, partly to make it better behaved and partly to adapt it to the STRESS function. Several problems have been solved so that the tunneling function does not lead to trivial solutions. Furthermore, we presented a minimization algorithm for the tunneling function based on majorization. The algorithm was extended to accommodate more than one pole, which was needed to avoid unwanted stationary points in the tunneling function. We extended the parametric programming algorithm to majorization. Several new majorization inequalities were found.

Two examples showed that our tunneling algorithm works, at least for these small problems. Experiments showed that a good value for the pole strength parameter is 1/3 and for the normalization is $n(n-1)/2$. In chapter 4, we consider a simulation study in which the tunneling method is compared to some other global optimization methods that were presented before.

uniformly distributed random coordinates in the interval [0, 1). The distances between the points of this matrix are normalized to have $\eta_\delta^2 = 1$. The $a\%$ error gauge is formed by multiplying each dissimilarity by $e^{a\varepsilon/100}$, where ε is standard normally distributed error (mean 0, variance 1). We normalize the gauge to have $\eta_\delta^2 = 1$. In Figure 4.1 we show the distribution of a particular small, a middle and a large sized dissimilarity to give an indication of the type of distribution that is obtained.

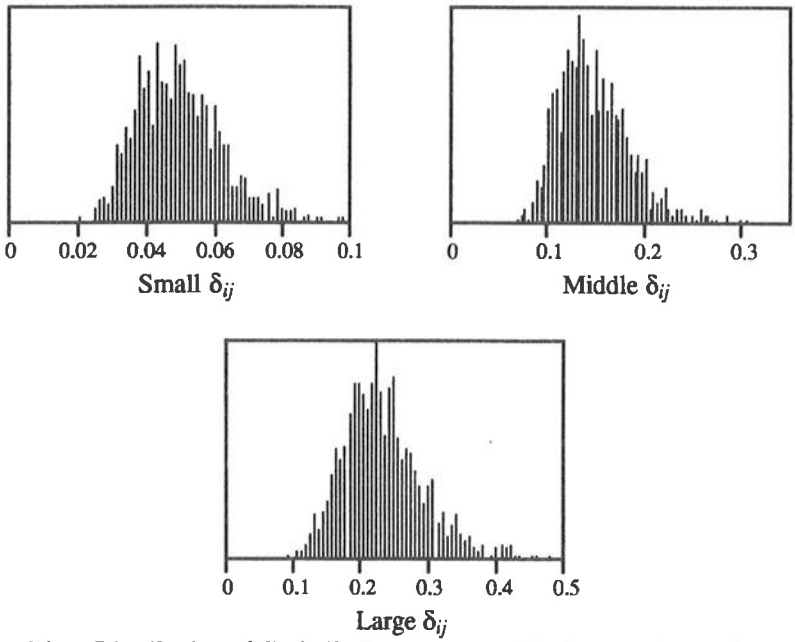


Figure 4.1 *Distribution of dissimilarity gauges used in the experiment, for a small, a middle and a large dissimilarity.*

The computations of the experiment were done on a SUN SPARC station in the interpreter language APL. For finding the number of different local minima we need to compare the local minimum configurations with each other. The comparison is complicated by the freedom of rotation. To decide whether two configurations are different, two strategies were suggested in section 2.5.2. The first one fixes some of the coordinates of all the configurations, so that the rotational freedom is removed. The second one compares the distances between the two configurations, rather than the coordinates. Implementations of the latter strategy turned out to be unsatisfactory for this experiment, because it was numerically not stable enough. Here, we used a third option. The configurations are rotated towards each other by using Procrustes rotation. For more details on Procrustes rotation we refer to Gower (1975) and Commandeur (1991). Here,

we regard two configurations equal if after Procrustes rotation the sum of squared differences between the coordinates is less than 0.01.

Table 4.1 *Results of the simulation study on the number of local minima in MDS. Reported are the number of different local minima found after 100 random starts and, in parentheses, the number of starts that ended in the lowest local minimum.*

Number of objects	Dimensionality	Error level 0 %	10 %	25 %	100%
10	2	5 (60)	5 (58)	2 (52)	2 (78)
	3	1 (100)	1 (100)	1 (100)	1 (100)
	5	1 (100)	1 (100)	1 (100)	1 (100)
	10	35 (1)	6 (20)	4 (61)	3 (42)
20	2	7 (9)	10 (2)	6 (84)	6 (17)
	3	2 (99)	3 (31)	1 (100)	3 (31)
	5	1 (100)	1 (100)	1 (100)	1 (100)
	10	1 (100)	1 (100)	3 (10)	1 (100)
40	2	11 (80)	1 (100)	6 (13)	10 (40)
	3	1 (100)	2 (99)	1 (100)	6 (24)
	5	1 (100)	1 (100)	3 (74)	1 (100)
	10	1 (100)	1 (100)	3 (50)	1 (100)
100	2	3 (98)	1 (100)	3 (98)	6 (91)
	3	1 (100)	1 (100)	1 (100)	2 (4)
	5	1 (100)	1 (100)	1 (100)	1 (100)
	10	1 (100)	1 (100)	1 (100)	1 (100)

The results of the experiment are presented in Table 4.1.We have to interpret this table with some care, since the experiment is limited in size and may depend on the accuracy chosen. The most striking fact from Table 4.1 is that for most combinations all 100 random startconfigurations ended in the candidate global minimum. Apparently, the SMACOF algorithm is quite capable in reaching the global minimum. Also, it seems that the current procedure for constructing gauges does not produce many local minima for moderate error levels. We also find that the number of different local minima is higher for small p than for higher dimensionality. This effect seems to be stronger for higher error levels. However, a different result was found for the combination $n = 10$ and $p = 10$, a case of full-dimensional scaling. Here, we found a high number of local minima. One reason for this could be that due to slow convergence of full-dimensional scaling the local searches may have stopped too early.

The current experiment was computationally very intensive. It took about 3 weeks CPU time to perform the entire experiment. One of the reasons for obtaining so few different local minima for large n might be that too few start configurations were used.

Obviously, with a larger number of start configurations like 1000 or 10000 we obtain a more accurate estimate of the number of local minima, although we expect that region of attraction to the candidate global minimum remains roughly the same. However, using so many start configurations for each of the combinations of error level, p, and n would have made the experiment impossible due to the computational effort.

This experiment lends support to the hypothesis that the local minimum problem is more severe for small dimensionality ($p = 2$, or $p = 3$) than for a larger dimensionality. Moreover, the amount of error imposed on the gauges led to more local minima, although this effect is less strong for larger n. Here, we have focused mainly on the number of local minima, but in section 4.1.3 we present some configurations of the local minima.

4.1.2 Performance of the tunneling method

In this section we investigate how the tunneling algorithm behaves for some of the difficult gauges of the previous section. A gauge is regarded as difficult if it has a moderate number of local minima, and a candidate global minimum with a small region of attraction.

The gauges that we use in this section stem from the experiment in the previous section (see Table 4.1) and have a small region of attraction to the candidate global minimum. The first gauge has $n = 20$, $p = 2$, and error level of 10%. We found 10 different local minima. Only two of the 100 local searches ended in the candidate global minimum. The second gauge has $n = 20$, $p = 10$, error level of 25%, 3 local minima, and 10 local searches out of 100 ended in the lowest local minimum. The third gauge has $n = 40$, $p = 2$, error level of 25%, 6 local minima, where 13 of the 100 local searches ended in the lowest local minimum. The tunneling method is used on these three rather difficult gauges.

We used the tunneling algorithm with multiple poles as defined in section 3.3. The pole strength parameter λ was set to 1/3 and the dissimilarities were normalized to have $\eta_\delta^2 = n(n-1)/2$, which are reasonable settings conform the fine tuning experiment in section 3.5. The tunneling step stops if a configuration is found with a smaller STRESS value than the previous local minimum. Otherwise, the tunneling step is stopped after 2000 iterations, or if the value of the auxiliary function $F(q,\mathbf{X})$ is less than 10^{-7}, which is an indication for a stationary point of the tunneling function. Then, at this position a pole is added to the tunneling function and the tunneling step is repeated. If more than 10 poles were needed, the tunneling step is regarded as having failed and the previous local minimum becomes the candidate global minimum. We started the tunneling algorithm from the worst local minimum. The STRESS values reported here are given in high accuracy, because the tunneling function is based on the STRESS values and hence is sensitive to the accuracy.

For gauge one, the tunneling method was started at a STRESS value of 9.68230903. The first tunneling step needed six poles to find a STRESS of 9.546557351. The next local

search yielded the candidate global minimum of 1.40219957. The CPU time needed was about 111 minutes. The tunneling method required six hours and 40 minutes to reach the candidate global minimum of the second gauge. The algorithm was started at a local minimum with STRESS value 5.30272142. Using at most 10 poles the tunneling method reached in three tunneling steps with local minimum STRESS values 5.30272077, 5.30263994, and 5.30263722, the candidate global minimum with STRESS 5.30263612. The small gains in STRESS could be due to the high dimensionality ($p = 10$) relative to the number of objects ($n = 20$) in this gauge. The third gauge imposed an even more difficult problem to the tunneling method, i.e., it needed 8 tunneling steps and about 19 hours of CPU time. The tunneling method was started at a STRESS of 89.31092138, which led in eight tunneling steps to 42.66376336 via the steps 89.29739523, 89.19337217, 89.14282561, 85.98388855, 85.82599954, 58.63553200, 42.66376437, and 42.66376435. Apparently there are at least three levels of local minima, one around STRESS values of 89, the second one around 85.8 and one around 42.6. This experiment suggests that for either large p (like gauge 2) and for large n (like gauge 3) the tunneling method is very computationally intensive. One of the reasons is that the update of tunneling step (3.54) requires the computation of an $n \times n$ inverse, which is of the complexity of the order n^3 (see also section 5.1.6). Another reason is that the tunneling method terminates only after the tunneling step has failed. This happens after 10 poles are added, which implies that at most 10 times 2000 iterations are performed. Therefore, the current termination criterion costs considerable computation time, especially when n is large.

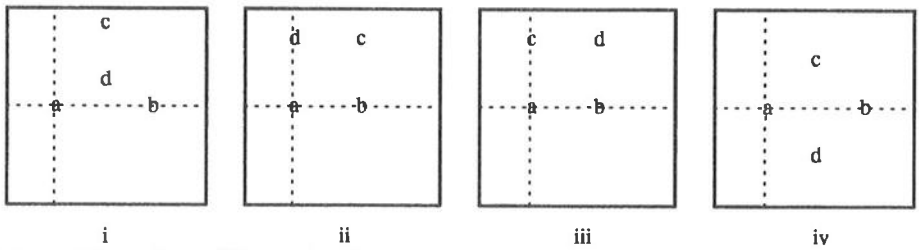
One of our conclusions from this experiment is that the tunneling method seems to be capable of finding even local minima with a small region of attraction. However, in its current implementation, it costs a huge amount of computation time, a feature that is shared with most other global optimization methods. Moreover, the tunneling method seems to be quite sensitive to the convergence criterion settings of the tunneling step. If for example the convergence criterion is set too weak, the maximum number of poles is not enough, or the maximum number of iterations is set too low, then the tunneling method will fail to arrive at a proper solution. Especially for middle or large sized n, say $n > 60$, the computational burden of the tunneling method seems to be prohibitive. For small MDS problems with or without a large region of attraction, the tunneling method could be applied readily.

4.1.3 Comparing multi-level-single-linkage and tunneling

Three different examples are presented to compare different aspects of multi-level-single-linkage clustering and tunneling. These results have been presented earlier in Groenen (1992). Here, we have a closer look at the different local minima themselves, to see in what respect they differ. The examples have a decreasing region of attraction to the global minimum, different number of objects, and one has a global minimum of zero. We

present for each example the results of MLSL with hypercubes, MLSL with moved hypercubes (see section 2.5), and of the tunneling method. Here, the dissimilarities are standardized to have sum of squares $n(n-1)/2$ and the configurations are of fixed dimensionality $p = 2$. The rotation invariance was solved here by fixing the first point in the configuration to be in the origin, and the second point to vary only on the nonnegative part of the first dimension (see section 2.5.2). The number of configurations drawn by MLSL is limited by the amount of memory available in the computer.

The first example is the 4×4 constant dissimilarity matrix discussed by De Leeuw (1988), see also section 3.4. He notices four types of stationary points, two of them in two dimensions: a configuration with three points at the corners of an equilateral triangle and the remaining one at the centroid ('triangle') and a configuration with four points at the corners of a square ('square'). MLSL was started with the triangle configuration. 4300 random configurations were drawn with N = 100, which induced a total of 124 local searches. With the exception of the first local search, all resulted in the square configuration, albeit in the three different forms given in Figure 4.2.



i ii iii iv

Figure 4.2 *Four different local minima of the constant dissimilarity matrix found by MLSL.*

Only one triangle was found by MLSL due to the initial configuration; the three others were not found. The triangle configuration is an example of a non-isolated stationary point (cf. De Leeuw, 1988). The square configuration has the least STRESS and a very large region of attraction, which suggests that it is the global minimum in two dimensions. Repeating the experiment, with moved hypercubes and the same sample size, decreased the number of local searches to 52 yielding the same local minima.

Tunneling was started with triangle configuration (i) of Figure 4.2 and with $\lambda = 1/3$. The start configuration of each tunneling step is a small perturbation of the previous local minimum. In 85 iterations the tunneling step yielded a configuration with the same STRESS as initial configuration and the subsequent local search yielded configuration ii of Figure 4.2. A plot of the history of iterations was already presented in Figure 3.2. The next tunneling step needed 307 iterations to emerge from the tunnel, after which the local search reached configuration iii of Figure 4.2. When both of these configurations were inserted as poles in $\tau_6(\mathbf{X})$ then tunneling was able to find the remaining square configuration.

In the second example, the dissimilarity matrix is the Euclidean distance matrix of a configuration of an equally spaced grid of nine points, see Figure 4.3 configuration (i). It shows what kind of local minima may be expected when the dissimilarities are Euclidean distances from a structured underlying configuration. MLSL generated 5800 configurations that were obtained from the uniform distribution. Using hypercubes 4070 local searches were started, which yielded 44 different local minima. The structure of the data yielded only 6 different species of local minima as shown in Figure 4.3. A species of local minima has one or more configurations that differ only in labelling of the points, not in the form of configuration. Almost 72% of the local searches ended in the global minimum indicating that it has a large region of attraction. With moved hypercubes, only 689 local were started from 5800 samples points, which led to 5 species of 22 local minima. Except for configuration (iii), the same species were found. Tunneling was started from the worst fitting configuration (vi) of Figure 4.3). After two tunneling steps it found the equally spaced grid of nine points.

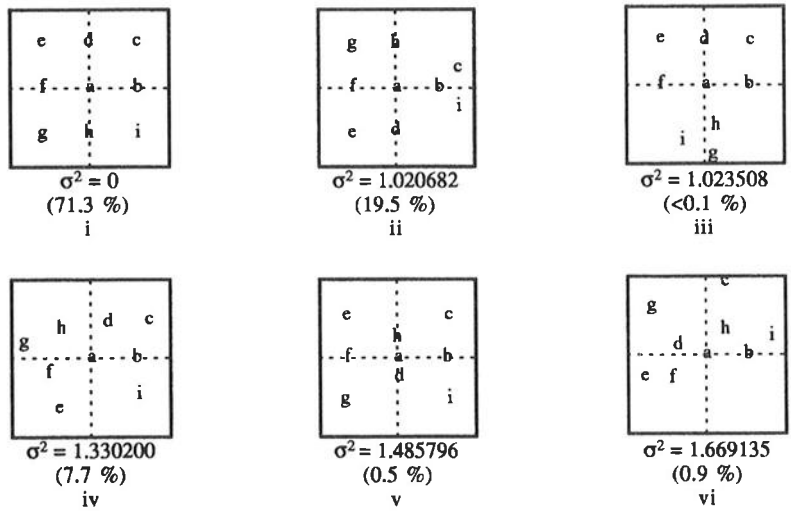


Figure 4.3 *The species of local minimum configurations obtained by MLSL of the Euclidean distance matrix of configuration i. Also given is their STRESS and the percentage of random points whose local search ended in this configuration*

The third example concerns genetic dissimilarities of bacterial strains that are reported in Mathar (1989) after experiments of Ihm (1986). Mathar (1991) obtained many different local minima using MDS on this data set, which makes it an interesting data set for the comparison of the two global optimization methods. The genetic dissimilarities were obtained as follows. Bacterial strains were investigated with respect to their phylogenetic similarity. DNA-sequences (chromosomes) of each two out of 17 different species have been brought together, heated and then annealed. If species are

similar then a large portion of equal parts of the individual DNA-sequences are associated in some way. This can be measured and gives the "percent-binding", which is a well-known term for biologists. The raw data were transformed to dissimilarities in the following way. If species i and j have a percent binding of a %, say, then $\delta_{ij} = 1 - a/100$.

The initial configuration of MLSL was the classical scaling solution (Torgerson, 1958; Gower, 1966) with STRESS 20.04914295. A local search starting from this solution resulted in a local minimum with STRESS 4.46047935.

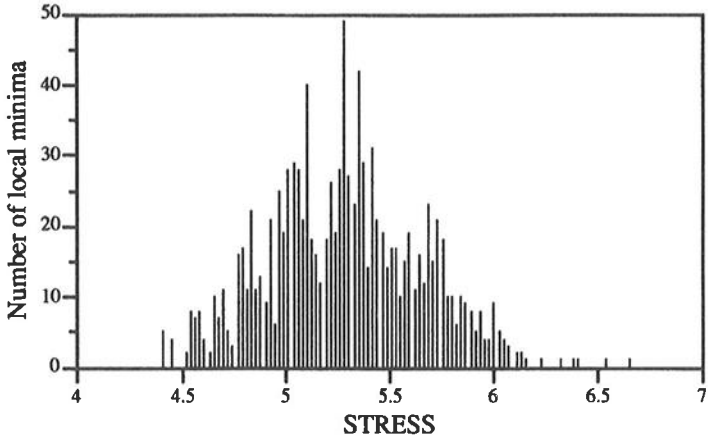


Figure 4.4 *Histogram of the number of local minima found with a certain STRESS value of the genetic dissimilarity data.*

Of 3200 random points that were drawn for MLSL with hypercubes, 3084 local searches were started that yielded 1098 different local minima. Although the convergence criterion of the local search procedure was strict (10^{-8}), we should not exclude that some of the 1098 minima may be regarded as the same. A histogram of the number of local minima with a certain STRESS value is presented in Figure 4.4. It shows that the majority of the local minima have rather high STRESS values. For this data set a global optimization method is clearly needed. In Figure 4.5 the five configurations with the lowest STRESS are given representing the local minima of 14% of the local searches. The differences between the configurations are small. Due to the rotation identification, configuration (ii) is more or less reflected along the second axis when compared to (i), but the main difference is the positioning of objects h, l and m. Comparing (i) with (iii), objects p and h are interchanged; (i) with (iv), the difference is in objects p, h and m; (i) with (v), the difference is in both the objects p, h, m and q, l. For this data set different local minima are formed by more or less interchanging the position of a few points. The experiment was repeated for moved hypercubes. With 6200 random points 29 local searches were needed, that resulted in 27 local minima, the best two being (i) and (iii) of Figure 4.5. Although only a very few local searches were started the assumed global minimum was found. The tunneling method was started with the worst local minimum

found by MLSL. After a series of 15 decreasing local minima and at most 7 poles, the tunneling method found configuration (i) of Figure 4.5. In fact, the last five local minima of tunneling coincided with the five configurations of Figure 4.5.

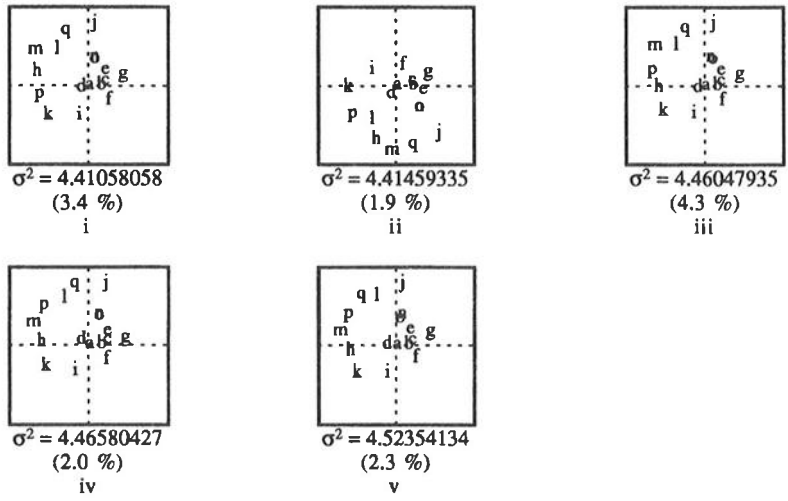


Figure 4.5 *The five best configurations of the genetic dissimilarity data obtained by MLSL, their STRESS and the percentage of random points whose local search ended in this configuration.*

It is interesting to see to what extent the local minima differ in their distances. After all, it is the distance matrix of a local minimum that determines the STRESS value. A two dimensional graphical representation can be obtained when using MDS as a meta analysis tool. To be more precise, we approximate the distance between the distance matrices of the local minima and obtain a low dimensional representation of the local minima. Thus the objects in the meta MDS analysis are the local minima. A dissimilarity δ_{ij} in the meta MDS analysis contains the distance between the distance matrix of local minimum i and that of local minimum j. Clearly, if two local minima have almost identical distance matrices, their distance in the meta MDS analysis should be small. Such a meta MDS analysis can be seen as a nonlinear approximation of a two dimensional plane that shows as much difference between the local minima as possible. Here, we took the best 100 local minima of the bacterial strain example and computed their distance matrix. Subsequently, each distance matrix was regarded as a point in $n(n-1)/2$ dimensional space. From these points a 100×100 meta distance matrix was constructed by computing the interpoint distance. Then, an ordinary MDS analysis in two dimensions was performed. The result is plotted in the horizontal plane of Figure 4.6, where vertically the STRESS value is reported. In the figure we find near the right side of the candidate global minimum a group of points with higher STRESS values of about 4.65. Other local minima with small STRESS are found at some further distance from the candidate global

minimum. This suggests that there is not a smooth valley of local minima leading to the candidate global minimum. It may indicate that it will be rather difficult to discover this minimum. In the figure we also see that the more distant we are from the candidate global minimum, the higher the STRESS values of the local minima tend to be. Thus, local minima that have a distance matrix that is rather different from the candidate global minimum distance matrix have high STRESS values.

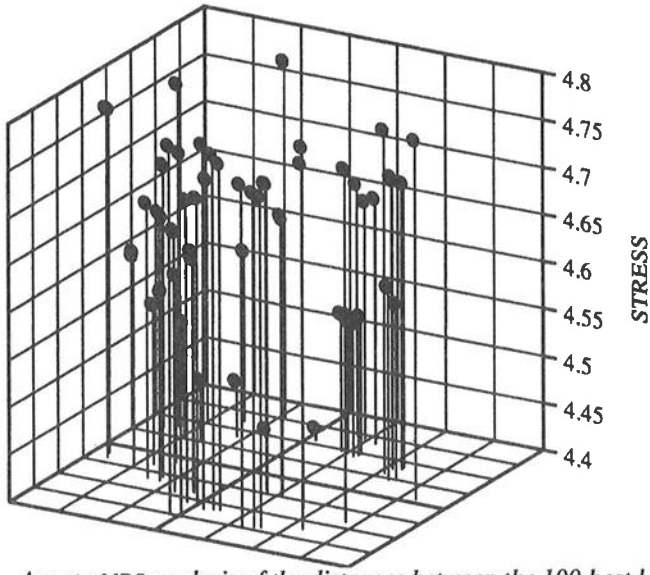


Figure 4.6 *A meta MDS analysis of the distances between the 100 best local minimum distance matrices of the bacterial strains example. In the horizontal plane the meta MDS solution is used, and the vertical axis gives the STRESS value for each local minimum.*

We now return to the comparison of MLSL and the tunneling method. From these examples we may conclude the following. The differences in the number of local searches of MLSL without moved hypercubes seems to depend on the number of parameters to be estimated, i.e., $n \times p$. In the first example with small n, relatively few local searches had to be performed as compared to the last two examples with larger n. Using moved hypercubes greatly reduced the number of local searches for all three examples, still yielding the same candidate global minimum. Therefore, we regard MLSL with moved hypercubes as a better version of multistart. In all examples, tunneling was successful in discovering the candidate global minimum, even though it has a small region of attraction in the bacterial strains example. The same example showed that the tunneling method still performs well, if there are many local minima. However, both MLSL and the tunneling method require heavy computations.

4.2 Numerical experiments for unidimensional scaling

The combinatorial nature of the unidimensional scaling problem makes it necessary to treat the numerical experiments separately from those for the multidimensional case, since optimal algorithms are basically different. To compare some of the different unidimensional scaling strategies (as discussed in section 2.2) in finding the global optimum, we performed the following experiment. First, we generate dissimilarity matrices with known optimal permutation. Then, we apply four pairwise interchange strategies, and the tabu search to see if the optimal permutation can be recovered. We compare their performance of reaching the global minimum and the effort that it takes. The dynamic programming approach is excluded from the experiment, because it always finds a global optimum and is therefore to be preferred if n is not too large.

In contrast to multidimensional scaling we are able for unidimensional scaling to generate a dissimilarity matrix that has known optimal configuration. In this way we can tell whether a unidimensional scaling strategy has reached a global minimum permutation or not. Hubert and Arabie (1986) showed that for fixed coordinates the unidimensional scaling problem can be viewed as a quadratic assignment problem, which is also a combinatorial problem. This feature can be exploited to generate dissimilarity matrices Δ with known (global) permutation. We use the test problem generator for Euclidean quadratic assignment problems as proposed by Li and Pardalos (1992). Their generator consists of the following four steps.

1. Compute a distance matrix $\mathbf{D}$ of a vector $\mathbf{x}$ with ordered coordinates, i.e., $x_1 < x_2 < \ldots < x_n$. Let N be the set $\{(i,j) \mid 1 \leq i < j \leq n\}$ of new pairs that are not updated yet. Set dissimilarities $\delta_{ij} = c$ (with $c > 0$) for $(i,j) \in N$ and $\delta_{ii} = 0$ for $1 \leq i \leq n$.
2. If the set N of new pairs is empty then stop.
3. Find pair (k,l) such that

$$d_{kl}(\mathbf{x}) = \max_{(i,j)\in N} d_{ij}(\mathbf{x}).$$

   If $l = k + 1$ then remove pair (k,l) from N and go to 2.
   Otherwise, choose randomly an integer s, with $k < s < l$ and draw ε from the uniform distribution between 0 and c. Update Δ by

$$\delta_{kl} = \delta_{kl} + \varepsilon,\ \delta_{ks} = \delta_{ks} - \varepsilon,\ \delta_{sl} = \delta_{sl} - \varepsilon.$$

4. Remove (k,l), (k,s) and (s,l) from N and go to step 2.

Li and Pardalos (1992) prove that the identity permutation remains the globally optimal permutation of the quadratic assignment problem. Therefore, the identity permutation is also the global minimum permutation of the unidimensional scaling problem. Note that the value of $t^2(\psi)$, that is maximized in unidimensional scaling, has the same value for dissimilarity matrices generated by the algorithm above with equal n and c. The

corresponding STRESS values can be different, because the sum of the squared dissimilarities η_δ^2 is usually different. Other test problem generators exist for this problem, but they are much more computationally intensive, which makes it difficult to generate large sized test problems.

In the following experiment, we only varied n, since the dimensionality $p = 1$. Error on the dissimilarities is implied by the test problem generator, and is thus excluded as a separate factor. We used the same four levels of n as in our multidimensional scaling experiment in section 4.1.1, i.e., n is 10, 20, 40, and 100. The pairwise interchange strategies (LOPI1, LOPI2, LOPI3, and LOPI4) and the tabu-search were started a hundred times with random start permutations. The search procedure in the tabu search was implemented with the LOPI1 pairwise interchange strategy. If the tabu step failed to reach a better function value within 100 iterations with a maximum of 20 tabu permutations, then the tabu step was terminated. For each of the 5 search procedures we report the number of times the identity permutation has been reached, the number of times that any permutation with global optimum value has been reached, and the average CPU time needed for reaching a minimum. Again the CPU time was measured on a SUN-SPARC workstation using the interpreter language APL. Therefore, the CPU time has to be interpreted with care, because some overhead may be caused by non-optimal coding of the interpreter especially for these programs. The results of the simulation study are presented in Table 4.2.

In the table, we see that the consecutive local pairwise interchange strategy, LOPI1, does not perform well when compared to the others. The LOPI2 strategy of Poole (1990), that searches for the best position for an object given the order of the other objects, is a good and relatively fast strategy in locating a global optimum. LOPI3, that does a pairwise interchange between any pair of objects if it increases $t^2(\psi)$, performs somewhat better, but costs more computing power. The LOPI4 strategy, that searches for the best pairwise interchange between over all object pairs, performs equally well as LOPI3, but needs dramatically more CPU time. For $n = 100$, LOPI4 was terminated after one week of computations. The tabu search in its current implementation did not perform much better than LOPI1, particularly for high n. Implementations of the tabu search based on better local search procedures (like LOPI2) might lead to better performance. A second improvement could be obtained by making the termination criterion of the tabu step dependent on n, i.e., terminate if after $10n$ iterations no better function value is found.

The current experiment indicates that Poole's (1990) LOPI2 strategy should be used to recover a global optimum with a reasonable probability and within moderate CPU time. If one wants to find a global minimum with a higher probability at the cost of extra computing time, then Heiser's (1989) LOPI3 strategy seems favourable. Clearly, if n is small enough, then the dynamic programming approach is always to be preferred, since it yields a global optimum by definition.

Table 4.2 *Simulation results of 5 strategies for unidimensional scaling.*

		Optimal $t^2(\psi)$	Identity permutation recovered (%)	Optimal VALUE $t^2(\psi)$ reached (%)	Average CPU time
$n = 10$	LOPI1	3.300	1	50	0.41
	LOPI2	3.300	16	83	1.59
	LOPI3	3.300	0	89	1.03
	LOPI4	3.300	2	88	2.00
	Tabu	3.300	1	61	1.77
$n = 20$	LOPI1	6.650	0	4	2.70
	LOPI2	6.650	32	75	10.30
	LOPI3	6.650	0	85	8.21
	LOPI4	6.650	0	89	35.80
	Tabu	6.650	0	9	9.81
$n = 40$	LOPI1	13.325	0	5	28.04
	LOPI2	13.325	0	97	40.79
	LOPI3	13.325	0	100	94.78
	LOPI4	13.325	0	99	845.37
	Tabu	13.325	0	7	40.72
$n = 100$	LOPI1	33.330	0	6	597.33
	LOPI2	33.330	0	100	560.13
	LOPI3	33.330	0	100	3124.54
	LOPI4	33.330	failed		
	Tabu	33.330	0	1	613.40

4.2.1 *Performance of tunneling method with unidimensional scaling*

As an illustration, we present a small example of the tunneling method for unidimensional scaling. Although for $p = 1$ the tunneling method does not use combinatorial structure, the procedure can be pursued just as if it concerned a multidimensional scaling problem. We use the example of the Kabah collection of archaeological deposits reported in Hubert and Arabie (1986), who took the example from Robinson (1951). The dynamic programming approach yielded a global minimum for the identity permutation with STRESS 4.73303773. The tunneling algorithm was started with the random permutation 2, 7, 5, 16, 17, 13, 15, 3, 1, 6, 8, 11, 12, 9, 14, 10, 4 that has STRESS 25.16402989. After 24 tunneling steps the algorithm arrived at a local minimum with STRESS 11.78850806 that has permutation 1, 2, 16, 17, 15, 13, 14, 12, 10, 11, 9, 8, 7, 6, 5, 4, 3. It is interesting to see that the relative order of many objects is almost correct, except for the positioning of objects 1 and 2. Apparently the tunneling

algorithm does not succeed in moving objects 1 and 2 to the other side of object 3. The tunneling method took about two hours and 15 minutes CPU time to arrive at the final permutation. Clearly, the combinatorial strategies of the previous section are much better to deal with unidimensional scaling. However, this small example does illustrate that the tunneling method still could be used even if $p = 1$.

4.3 Conclusions

We have treated the local minimum problem differently for unidimensional scaling and multidimensional scaling. For multidimensional scaling, our experiment in section 4.1.1 suggests that increasing error enlarges the number of local minima. We also found less local minima for higher dimensionality with respect to the number of objects. The tunneling algorithm is equally capable of finding the candidate global minimum as is multistart or MLSL clustering. However, all these methods are computationally very demanding. The settings of the tunneling method (like maximum number of poles, maximum number of iterations in a tunneling step, convergence criterion of the tunneling function) need to be strong not to fail. For MLSL clustering the use of moved hypercubes greatly reduces the number of local searches, and still recovered the same candidate global minimum in our experiment.

For unidimensional scaling the first choice should be the dynamic programming approach, since it guarantees a global optimum. For large n, say $n \geq 20$, the dynamic programming strategy is no longer feasible, so that we have to revert to other strategies, like pairwise interchange. These local pairwise interchange strategies and the tabu search were compared on their capability in recovering a global optimum that was generated using an algorithm of Li and Pardalos (1992). In fact, the LOPI2 strategy of Poole (1990) and the LOPI3 strategy of Heiser (1989) give a good performance within a reasonable amount of CPU time. The LOPI1 and LOPI4 strategies did not perform very favourably. The former is fast, but often does not locate the global optimum, whereas the latter is able to find the global optimum, but takes a huge amount of CPU time. The tabu search performed only a little bit better than LOPI1, but not as good as the other pairwise interchange strategies. A small example showed that the tunneling algorithm still works for unidimensional scaling, but is not successful in finding the global optimum.

CHAPTER 5

INCOMPLETE MDS

An important feature of the STRESS function is the inclusion of weights. One of the advantages of having a weight for each pair of objects is that missing data are handled more easily. A zero weight indicates absence of the dissimilarity and a non-zero weight the presence. However, using weights most often requires extra computational effort. Here, we shall look at specially structured designs, for which these computations can be simplified. As a second advantage, it permits us to formulate MDS models, for which the set of objects is split into two mutually exclusive sets. Within this framework we discuss unfolding, external unfolding and (semi-)complete scaling. Furthermore, we apply semi-complete MDS in a new method, called moving frame MDS. We present numerical examples to see if moving frame MDS accelerates the computations of complete MDS.

5.1 Structured designs in MDS

For large dissimilarity tables in MDS two problems occur. The first problem consists of the excessive number of dissimilarities to be gathered, since the total number of different pairs of stimuli equals $n(n\text{-}1)/2$, where n is the number of stimuli. The second problem is how to analyse them efficiently. Both problems can be handled by using incomplete designs; i.e., each dissimilarity is weighted by zero or one indicating its presence or absence in the incomplete design. The subset of dissimilarities in the incomplete design thus reduces the number of dissimilarities to be collected. As well as simplifying the respondents' task, theoretical considerations suggest that incomplete designs can also be used to accelerate computations even when all dissimilarities might be readily available (or calculable). Spence and Domoney (1974) found a good recovery of the target configuration by using designs that required only a small fraction of all possible dissimilarities. The SMACOF algorithm needs the inverse of a square matrix of the order of the number of stimuli, see section 1.3. As n gets large, computation of the inverse may become a burden. For some structured designs Gower and Groenen (1991) and Groenen (1993) discussed how the computation of such an inverse can be accelerated by making use of arguments of the modified Leverrier–Faddeev algorithm. We summarize their results in the following sections.

An incomplete design can be described by the $n \times n$ matrix $\mathbf{W}$ where the weights $w_{ij} = w_{ji}$, are restricted to be zero or one and w_{ii} is zero for all $i = 1, ..., n$. In section 1.3 it was outlined that the SMACOF algorithm needs the Moore-Penrose inverse of

$$\mathbf{V} = \text{diag}(\mathbf{1}'\mathbf{W}) - \mathbf{W}. \tag{5.1}$$

If we assume that $\mathbf{V}$ is irreducible, then $\mathbf{V}$ is of rank $n-1$, since $\mathbf{V}$ has row and column sums equal to zero and thus has the vector $\mathbf{1}$ in the null-space. Accordingly, the Moore-Penrose inverse $\mathbf{V}^{-}$ is given by $(\mathbf{V} + \mathbf{1}\mathbf{1}')^{-1} - n^{-2}\mathbf{1}\mathbf{1}'$. If all weights are equal to one, then $\mathbf{V}$ equals n times the centering operator, i.e., $\mathbf{V} = n\mathbf{I} - \mathbf{1}\mathbf{1}'$ and $\mathbf{V}^{-}$ equals $n^{-1}(\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}')$. In SMACOF, $\mathbf{V}^{-}$ is multiplied by a matrix that has zero column means and therefore we only have to find $(\mathbf{V} + \mathbf{1}\mathbf{1}')^{-1}$. To simplify the computation of the inverse, we use a combination of standard conditions on the inverse and arguments derived from the modified Leverrier-Faddeev algorithm.

We discuss three classes of structured designs, which are displayed in Figure 5.1. The first class is formed by the partitioned block designs that partition the dissimilarity matrix into non-overlapping blocks, only some of which are to be used. The second class is formed by the block tridiagonal designs, which is a special case of the partitioned block designs. The third class we discuss are the (block) circular designs, where the matrix to be inverted is a circulant matrix.

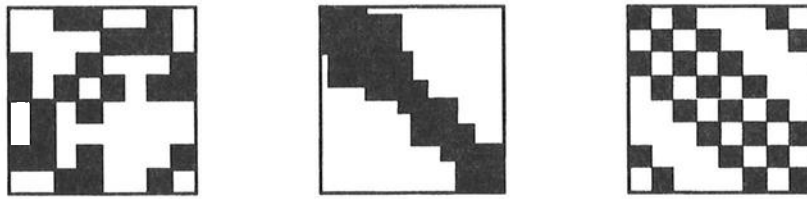


Figure 5.1 *Examples of a partitioned block design (a), a block tridiagonal design (b) and a circular design (c).*

The structure in these three classes of designs is exploited by the modified Leverrier Faddeev algorithm to make the computation of the inverse more efficient. A simulation study of partitioned block designs is presented to illustrate the gain in efficiency.

5.1.1 The modified Leverrier–Faddeev algorithm

The modified Leverrier–Faddeev algorithm can be used to find the inverse or the spectral decomposition of a matrix. The actual numerical algorithm may be used, but Gower

(1980) warns that the "... algorithm is inefficient and inaccurate and is clearly unsuitable for numerical work". A strong feature of the algorithm lays in the theoretical arguments that can be derived from the modified Leverrier–Faddeev algorithm, from which an explicit spectral decomposition or an explicit inverse of patterned matrices may be obtained. Here, we briefly explain the algorithm and show how to simplify the computation of $\mathbf{A}^{-1} = (\mathbf{V} + \mathbf{11}')^{-1}$ by making use of the patterns arising from structured incomplete designs.

The Leverrier-Faddeev algorithm can be summarized by the sequence

$$\begin{aligned} \mathbf{Y}_0 &= \mathbf{A} \\ \mathbf{Y}_i &= \mathbf{A}\mathbf{Y}_{i-1} + p_i\mathbf{A}, \qquad i = 1,\ldots,n \end{aligned} \tag{5.2}$$

where p_i equals $-\frac{1}{i}\mathrm{tr}\mathbf{Y}_{i-1}$ which is one of the coefficients of the characteristic polynomial of **A**. The inverse is given by

$$\mathbf{A}^{-1} = -(1/p_n)(\mathbf{Y}_{n-2} + p_{n-1}\mathbf{I}) \tag{5.3}$$

and similar expressions exist for the Moore-Penrose inverse and the eigenvectors (Gower, 1980). This algorithm breaks down when **A** has repeated eigenvalues but a modified version in which p_i is replaced by q_i, the coefficients of the minimal polynomial of **A**, remains valid, except under unusual circumstances that are irrelevant to the current discussion. The main characteristic of this algorithm that we use here is the following. Suppose matrix **A** and **Y** are similarly patterned matrices and suppose that **AY** and **A** + **Y** result in a matrix of the same pattern. Then, the modified Leverrier–Faddeev algorithm implies that the inverse of **A** shares the same pattern. This can be seen by inductive reasoning and by inspecting (5.2) closely. Suppose **A** meets the assumptions and so does $\mathbf{Y}_{i-1}$, then $\mathbf{A}\mathbf{Y}_{i-1} + p_i\mathbf{A}$, yields a similar patterned matrix. The condition holds by definition for $\mathbf{Y}_0$. Therefore each $\mathbf{Y}_i$ has the same pattern as **A**, and consequently so has $\mathbf{A}^{-1}$.

5.1.2 Partitioned block designs

In a partitioned block design the objects can be partitioned in k groups of size n_1, n_2, ..., n_k. All the elements in the $k \times k$ blocks ij of size $n_i \times n_j$ have equal values, either zero or one, say ξ_{ij}. Consequently the matrix **A**, equal to $\mathbf{V} + \mathbf{11}'$, has associated blocks $\xi_{ij}\mathbf{11}'$ with ξ_{ij} equal to one if a block is absent and ξ_{ij} equal to zero if a block is present. Note that throughout this section **11'** and **I** are conformable to the block sizes of ij. The pattern of the partitioned block design is reflected in **A**, which has the form

$$\mathbf{A} = \begin{bmatrix} \gamma_1\mathbf{I}_1 & & & \\ & \gamma_2\mathbf{I}_2 & & \\ & & \ddots & \\ & & & \gamma_k\mathbf{I}_k \end{bmatrix} + \begin{bmatrix} \xi_{11}\mathbf{1}\mathbf{1}' & \xi_{12}\mathbf{1}\mathbf{1}' & \cdots & \xi_{1k}\mathbf{1}\mathbf{1}' \\ \xi_{21}\mathbf{1}\mathbf{1}' & \xi_{22}\mathbf{1}\mathbf{1}' & \cdots & \xi_{2k}\mathbf{1}\mathbf{1}' \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{k1}\mathbf{1}\mathbf{1}' & \xi_{k2}\mathbf{1}\mathbf{1}' & \cdots & \xi_{kk}\mathbf{1}\mathbf{1}' \end{bmatrix} \tag{5.4}$$

where because every row of **A** has sum n, $\gamma_i = \sum_s(1-\xi_{is})n_s$. Let **Z** be a matrix with a similar pattern as **A**, defined by the diagonal blocks $\theta_i\,\mathbf{I}_i$ and $\varphi_{ij}\mathbf{1}\mathbf{1}'$. The multiplication **AZ** results in a similar patterned matrix, with block ij given by

$$(\gamma_i\varphi_{ij} + \theta_j\xi_{ij} + \sum_{s=1}^{k} n_s\xi_{is}\varphi_{sj})\mathbf{1}\mathbf{1}' \qquad \text{for } i \neq j \tag{5.5}$$

$$(\gamma_i\varphi_{ii} + \theta_i\xi_{ii} + \sum_{s=1}^{k} n_s\xi_{is}\varphi_{si})\mathbf{1}\mathbf{1}' + \gamma_i\,\theta_i\mathbf{I}_i \qquad \text{for } i = j. \tag{5.6}$$

Thus, the multiplication of two partitioned block matrices retains the pattern. It is easy to see that the result of **A** + **Z** also keeps the pattern. As a consequence of the Leverrier–Faddeev algorithm, $\mathbf{A}^{-1}$ shares this same pattern. This important feature allows us to compute the inverse of **A** more efficiently. In the following, let **Z** be the inverse of **A**, so that **AZ** must equal **I**. Therefore (5.5) must be zero, and the coefficient of $\mathbf{1}\mathbf{1}'$ in (5.6) must also be zero. Further it follows directly that $\gamma_i\theta_i\mathbf{I}_i$ is equal to $\mathbf{I}_i$ which implies θ_i equals $1/\gamma_i$. Thus

$$\gamma_i\varphi_{ij} + \sum_{s=1}^{k} n_s\xi_{is}\varphi_{sj} = -\theta_j\xi_{ij} \tag{5.7}$$

must hold for all pairs ij. This set of $k\times k$ equations can be rewritten as $\mathbf{\Phi N A}_r = -\mathbf{\Theta\Xi}$, where **N** is a diagonal matrix with elements n_i and $\mathbf{A}_r$ equals $(\mathbf{\Gamma N}^{-1}+\mathbf{\Xi})$. The only matrix not known is $\mathbf{\Phi}$, which is equal to $-\mathbf{\Theta\Xi A}_r^{-1}\mathbf{N}^{-1}$. Instead of computing the $n\times n$ inverse of $\mathbf{V} + \mathbf{1}\mathbf{1}'$ it suffices to compute the $k\times k$ inverse of $\mathbf{A}_r$ and do some matrix multiplications of order k. As a consequence, a considerable gain in speed can be expected especially when k is small. A simulation study to corroborate this claim is presented in the next section.

5.1.3 Block tridiagonal designs

A fast computational method for the inverse of block tridiagonal designs can be formulated by making use of the structured sparseness of $\mathbf{\Xi}$. Such a design consists of blocks on the diagonal and both adjacent subdiagonals. In this case, the elements of $\mathbf{\Xi}$ equal zero or one, indicating whether the block is present or absent. The second and higher subdiagonals of $\mathbf{A}_r$ have unit values. As a first step towards finding the inverse of $\mathbf{A}_r$ it is convenient to subtract the rank one matrix $\mathbf{1}\mathbf{1}'$ to obtain a sparse tridiagonal matrix

$$\mathbf{B} = \mathbf{A}_r - \mathbf{1}\mathbf{1}' = \begin{bmatrix} \beta_1 & -1 & \cdots & 0 & 0 \\ -1 & \beta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \beta_{k-1} & -1 \\ 0 & 0 & \cdots & -1 & \beta_k \end{bmatrix} \tag{5.8}$$

with $\beta_1 = n_2/n_1$, $\beta_i = (n_{i-1}+n_{i+1})/n_i$ and $\beta_k = n_{k-1}/n_k$. $\mathbf{B}$ is more sparse than $\mathbf{A}_r$ for k larger than 3. However, because $\mathbf{Bn} = \mathbf{0}$, the matrix $\mathbf{A}_r = \mathbf{B} + \mathbf{1}\mathbf{1}'$ cannot be inverted from knowledge of $\mathbf{B}^{-1}$. To modify $\mathbf{B}$ to be of full rank, while retaining its tridiagonal form, we may proceed as follows. Let $\mathbf{e}$ be the unit vector that has zero values everywhere except for its final element which has value one. Then for any valid design $\mathbf{e}'\mathbf{n} \neq 0$ and we may write:

$$\mathbf{A}_r = \mathbf{1}\mathbf{1}' - \mathbf{e}\mathbf{e}' + \mathbf{T} \tag{5.9}$$

where $\mathbf{T} = \mathbf{B} + \mathbf{e}\mathbf{e}'$. $\mathbf{T}$ is tridiagonal and of full rank, with $\mathbf{Tn} = (\mathbf{e}'\mathbf{n})\mathbf{e}$, so that

$$\mathbf{T}^{-1}\mathbf{e} = \mathbf{n}/(\mathbf{e}'\mathbf{n}) \ . \tag{5.10}$$

Using result (5.10) and after some manipulations:

$$\mathbf{A}_r^{-1} = \left(\mathbf{I} - \frac{\mathbf{n}\mathbf{1}'}{\mathbf{1}'\mathbf{n}}\right)\mathbf{T}^{-1}\left(\mathbf{I} - \frac{\mathbf{1}\mathbf{n}'}{\mathbf{1}'\mathbf{n}}\right) + \frac{\mathbf{n}\mathbf{n}'}{(\mathbf{1}'\mathbf{n})^2} \ . \tag{5.11}$$

Thus, to invert $\mathbf{A}_r$ it is only necessary to invert the tridiagonal matrix $\mathbf{T}$. The Cholesky decomposition $\mathbf{LDL}'$ of $\mathbf{T}$ may be used to find $\mathbf{T}^{-1}$, where the lower triangular matrix $\mathbf{L}$ is

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -1/d_1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & -1/d_{k-1} & 1 \end{bmatrix}$$

and the diagonal matrix $\mathbf{D}$ has non-zero elements $d_1 = \beta_1$, $d_i = \beta_i - d_{i-1}^{-1}$ for $i = 2, 3, \ldots, k-1$ and $d_k = 1$. The inverse of $\mathbf{T}$ is given by $(\mathbf{L}^{-1})'\mathbf{D}^{-1}\mathbf{L}'^{-1} = \mathbf{M}'\mathbf{D}^{-1}\mathbf{M}$, where the lower triangular matrix $\mathbf{M}$ has diagonal elements unity and subdiagonal elements

$$m_{ij} = \left(\prod_{s=j}^{i-1} d_s\right)^{-1} \ . \tag{5.12}$$

One important characteristic of the block tridiagonal design is that the associated inverse can be computed relatively fast. The gain in speed when using this method depends on the number of blocks as well as their size, but the gain can be considerable.

5.1.4 Circular designs

An important class of designs is when Ξ denotes a symmetric circulant design (see Spence, 1982). **A** is a circulant if each subsequent row equals the previous one, provided that all elements are shifted one position to the right and the final element is put in the first position. If **A** is also symmetric then we call **A** a symmetric circulant. Thus, the first row equals $(a_1, a_2, a_3, ..., a_m, ..., a_3, a_2)$, the second $(a_2, a_3, a_4, ..., a_{m-1}, ..., a_4, a_3)$ and so on. A circulant is symmetric if and only if $a_j = a_{n-j+2}$ for all $j = 1, 2,..., n$. Note that the central value a_m is repeated twice if n is odd, where $m = (n + 1)/2$ for n odd and $m = (n + 2)/2$ for n even. Both matrix multiplication and addition of two symmetric circulant matrices result in a symmetric circulant matrix. As a consequence of the modified Leverrier–Faddeev algorithm, the (Moore-Penrose) inverse must also be a symmetric circulant. Gower and Groenen (1991) gave an explicit result for the inverse of a symmetric circulant based on the spectral decomposition. A symmetric circulant **A** has eigenvalues

$$\lambda_j = \lambda_{n-j} = a_1 + \kappa(-1)^j a_m + 2\sum_{k=2}^{m-\kappa} a_k \cos\big((k-1)j\theta\big), \tag{5.13}$$

and inverse $\mathbf{A}^{-1}$ is a symmetric circulant with elements $n^{-1}\mathbf{c}$, where

$$c_k = \lambda_n^{-1} + \kappa(-1)^{\kappa+1}\lambda_{m-1}^{-1} + 2\sum_{j=1}^{m-\kappa-1} \lambda_j^{-1} \cos\big((k-1)j\theta\big), \tag{5.14}$$

and where $\theta = 2\pi$, $\kappa = 0$ when n is odd and $\kappa = 1$ when n is even. Note that Spence (1982) reported the eigenvalues (5.13) which he used to express the efficiency of circular designs.

For a general partitioned block design, $\mathbf{A}_r$ will not be a symmetric circulant because the diagonal values of $\Gamma\mathbf{N}^{-1}$ are not constant and then $\mathbf{A}_r$ seems to need full inversion. However, there are two important cases where advantage can be taken of the explicit result of (5.14). These are (i) when $\mathbf{n} = n/k\,\mathbf{1}$, which requires that n is a multiple of k, and (ii) when $n_s = 1$ for all s, so that $\alpha_i = \sum_s (1-\xi_{is})$, which is a constant because the rows of the circulant Ξ have constant sum. Case (i) merely requires the inverse of the $k\times k$ symmetric circulant $\mathbf{A}_r$. Case (ii) is especially important because, although the block-structure is lost (all the blocks are of size 1×1), the ξ's may be chosen such that Ξ is a symmetric circulant with $\boldsymbol{\xi}'$ equal to (1,1,0,...,0,...,0,1), which gives a tridiagonal design with a single element in positions $(1,n)$ and $(n,1)$ or we may choose a band design a symmetric circulant Ξ with $\boldsymbol{\xi}'$ = (1,1,1,...,1,0,...,0,...,0,1,...,1,1) supplemented by extra

cells in the upper-right and lower-left corner. Other choices such as Ξ being a symmetric circulant with $\xi' = (1,1,0,1,0,...,0,...,0,1,0,1)$ determine a more interlocking type of design. Note that, because **1** is an eigenvector of every circulant matrix, we may operate on $\mathbf{A} - \mathbf{11}'$ and then, omitting the term in λ_n^{-1}, (5.14) gives the generalized inverse required by MDS. All such designs may be analyzed very efficiently by the method presented here, especially when many of the ξ's are zero.

5.1.5 Simulation results

A simulation study was performed to evaluate the gain of speed in finding the inverse of $\mathbf{V} + \mathbf{11}'$. The study was limited to partitioned block designs only, because it is potentially the most computationally intensive task, since it requires the calculation of a $k \times k$ inverse. For block tridiagonal and circular designs this $k \times k$ inverse can be solved explicitly as outlined above. The speed of computing the inverse of several partitioned block design matrices was evaluated for each of two conditions, one using the information available from partitioned block designs, condition 1, (thus inverting a $k \times k$ matrix) and the other computing the full $n \times n$ inverse of $\mathbf{V} + \mathbf{11}'$, condition 2. Furthermore, the size of the matrix was varied ($n = 10, 25, 50, 100, 250$) and the number of blocks was varied ($k = 2, 4, 5, 10, 25$). For each combination the CPU time needed in FORTRAN77 to find the inverse on a SUN SPARC station for both conditions was recorded. Ordinary matrix inverses were computed by LU decomposition. The average CPU time for both conditions are reported in Figure 5.2 for each of the five different sizes of the matrix. The average CPU time for each number of blocks for the two conditions can be seen in Figure 5.3. In both figures the average CPU time is reported on a log scale.

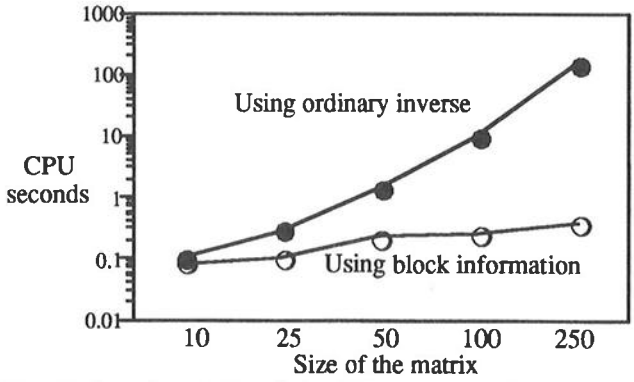


Figure 5.2 *The number of CPU seconds it takes to compute the inverse of a partitioned block design for various sizes of the design (averaged over the different number of blocks).*

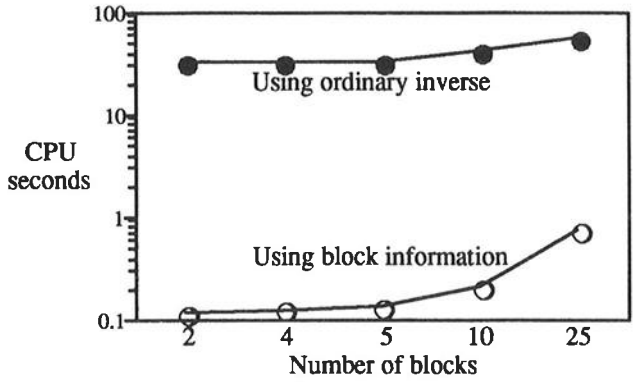


Figure 5.3 *The number of CPU seconds it takes to compute the inverse of a partitioned block design for various number of blocks (averaged over the size of the matrix).*

We clearly see that for small sized n there is not so much difference between the two conditions. However, a practical gain in speed is obtained for larger sized matrices (say n larger than 25). For example, the average CPU time needed to find the inverse of the 250×250 design without using block information (condition 2) was around 148 seconds, in contrast to 0.37 seconds for condition 1. As the number of blocks increases condition 1 outperforms condition 2 by far, though the average CPU time of condition 1 increases slowly too.

It is clear that condition 1 yields a significantly faster computation of the inverse required for a partitioned block design than condition 2. This gain is of particular importance for large sized designs.

5.1.6 Discussion and conclusions

The arguments from the modified Leverrier–Faddeev algorithm allow us to gain efficiency in the computation of the inverse needed in MDS when using partitioned block designs, block tridiagonal designs and circular designs. The simulation study for partitioned block designs shows that this tailor-made method results in a considerable decrease of computing time in finding the inverse. This is not surprising, because inverting by LU decomposition is a process that is cubic in n (see Press, Flannery, Teukolsky, and Vetterling, 1988). Thus, if the size of the matrix to be inverted can be reduced to k, then the computation time can be expected to be of the order $(n/k)^3$ times faster. However, it must be noted that the SMACOF algorithm for MDS is an iterative procedure that, in ordinary cases, needs to compute this inverse only once. Consequently, the gain in speed is effective only once. The structure of the design can also be exploited in computing the Guttman transform more efficiently by disregarding blocks that have zero weight, thus obtaining an increase in speed in every iteration.

5.2 Multidimensional scaling with two sets of objects

For some applications of MDS it is useful to partition the set of objects into two mutually exclusive groups. By making use of special weights and keeping the coordinates in one of the sets fixed, some known and unknown models can be obtained, i.e., (external) unfolding, semi-complete MDS.

Let us assume that the first n_1 objects form group 1 and the last n_2 objects form group 2. Note that if the objects in group 1 were not consecutive, we could always permute both the objects and the dissimilarities, such that they become consecutive. Remember from chapter 1 that STRESS can be written as

$$\sigma^2(\mathbf{X}) = 1 + \mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{X})\mathbf{X} = 1 + \eta^2(\mathbf{X}) - 2\rho(\mathbf{X}), \tag{5.15}$$

where for convenience η_δ^2 is normalized to one without loss of generality. By majorization it was shown that (5.15) is always smaller than

$$\hat{\sigma}^2(\mathbf{X},\mathbf{Y}) = 1 + \mathrm{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\mathrm{tr}\mathbf{X}'\mathbf{B}(\mathbf{Y})\mathbf{Y} = 1 + \eta^2(\mathbf{X}) - 2\hat{\rho}(\mathbf{X},\mathbf{Y}), \tag{5.16}$$

which is minimized by taking the Guttman transform $\mathbf{X}^+ = \mathbf{V}^-\mathbf{B}(\mathbf{Y})\mathbf{Y}$.

Since the stimuli fall apart into two groups, $\eta^2(\mathbf{X})$ and $\hat{\rho}(\mathbf{X},\mathbf{Y})$ may also be written in terms of partitioned matrices $\mathbf{X}_1$ and $\mathbf{X}_2$. Thus, $\eta^2(\mathbf{X})$ can be written as

$$\begin{aligned}\eta^2(\mathbf{X}) &= \mathrm{tr}\begin{bmatrix}\mathbf{X}_1' & \mathbf{X}_2'\end{bmatrix}\begin{bmatrix}\mathbf{V}_{11} & \mathbf{V}_{12}\\ \mathbf{V}_{12}' & \mathbf{V}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{X}_1\\ \mathbf{X}_2\end{bmatrix} \\ &= \mathrm{tr}\,\mathbf{X}_1'\mathbf{V}_{11}\mathbf{X}_1 + \mathrm{tr}\,\mathbf{X}_2'\mathbf{V}_{22}\mathbf{X}_2 + 2\,\mathrm{tr}\,\mathbf{X}_1'\mathbf{V}_{12}\mathbf{X}_2\end{aligned} \tag{5.17}$$

and $\hat{\rho}(\mathbf{X},\mathbf{Y})$ may be expressed as

$$\begin{aligned}\hat{\rho}(\mathbf{X},\mathbf{Y}) &= \mathrm{tr}\begin{bmatrix}\mathbf{X}_1' & \mathbf{X}_2'\end{bmatrix}\begin{bmatrix}\mathbf{B}_{11}(\mathbf{Y}) & \mathbf{B}_{12}(\mathbf{Y})\\ \mathbf{B}_{12}(\mathbf{Y})' & \mathbf{B}_{22}(\mathbf{Y})\end{bmatrix}\begin{bmatrix}\mathbf{Y}_1\\ \mathbf{Y}_2\end{bmatrix} \\ &= \mathrm{tr}\,\mathbf{X}_1'\mathbf{B}_{11}(\mathbf{Y})\mathbf{Y}_1 + \mathrm{tr}\,\mathbf{X}_1'\mathbf{B}_{12}(\mathbf{Y})\mathbf{Y}_2 \\ &\quad + \mathrm{tr}\,\mathbf{X}_2'\mathbf{B}_{22}(\mathbf{Y})\mathbf{Y}_2 + \mathrm{tr}\,\mathbf{X}_2'\mathbf{B}_{12}(\mathbf{Y})'\mathbf{Y}_1.\end{aligned} \tag{5.18}$$

Note that the assumption of irreducibillity of $\mathbf{W}$ from chapter 1 implies $\mathbf{V}_{12} \neq \mathbf{0}$. Combining (5.17) and (5.18) gives

$$\begin{aligned}\sigma^2(\mathbf{X}) \leq \hat{\sigma}^2(\mathbf{X},\mathbf{Y}) = 1 &+ \mathrm{tr}\mathbf{X}_1'\mathbf{V}_{11}\mathbf{X}_1 - 2\mathrm{tr}\mathbf{X}_1'\mathbf{B}_{11}(\mathbf{Y})\mathbf{Y}_1 \\ &+ \mathrm{tr}\mathbf{X}_1'\mathbf{V}_{12}\mathbf{X}_2 - 2\mathrm{tr}\mathbf{X}_1'\mathbf{B}_{12}(\mathbf{Y})\mathbf{Y}_2 \\ &+ \mathrm{tr}\mathbf{X}_2'\mathbf{V}_{22}\mathbf{X}_2 - 2\mathrm{tr}\mathbf{X}_2'\mathbf{B}_{22}(\mathbf{Y})\mathbf{Y}_2 \\ &+ \mathrm{tr}\mathbf{X}_2'\mathbf{V}_{12}\mathbf{X}_1 - 2\mathrm{tr}\mathbf{X}_2'\mathbf{B}_{12}(\mathbf{Y})'\mathbf{Y}_1.\end{aligned} \tag{5.19}$$

To obtain some interesting models, we introduce four factors that vary for different models in MDS with two groups of objects. First, $\mathbf{W}_{11}$ can have zero weights everywhere, so that the dissimilarities of objects within group 1 are missing. Second, the dissimilarities within the objects of group 2 are missing, i.e., $\mathbf{W}_{22}$ equals zero. Thirdly, provided that $\mathbf{X}_1$ is updated, we may keep $\mathbf{X}_2$ fixed or we may leave it free. Some additional special cases occur if we distinguish between several group sizes of $\mathbf{X}_1$, i.e., $n_1 = 1$, $1 < n_1 < n - 1$ and $n_1 = n - 1$. Below, we discuss the models that can be formed using these options. A summary is given in Table 5.1.

Table 5.1 *Summary of different settings in MDS with two groups of objects.*

		$n_1 = 1$	$1 < n_1 < n - 1$	$n_1 = n - 1$
Set II fixed				
$\mathbf{W}_{11}=\mathbf{0}$	$\mathbf{W}_{22}=\mathbf{0}$	cyclic update	external unfolding	trivial
	$\mathbf{W}_{22}\neq\mathbf{0}$	cyclic update	external unfolding	trivial
$\mathbf{W}_{11}\neq\mathbf{0}$	$\mathbf{W}_{22}=\mathbf{0}$	cyclic update	semi complete MDS	fixed point MDS
	$\mathbf{W}_{22}\neq\mathbf{0}$	cyclic update	semi complete MDS	fixed point MDS
Set II free				
$\mathbf{W}_{11}=\mathbf{0}$	$\mathbf{W}_{22}=\mathbf{0}$	trivial	unfolding	trivial
	$\mathbf{W}_{22}\neq\mathbf{0}$	complete MDS	almost complete MDS	trivial
$\mathbf{W}_{11}\neq\mathbf{0}$	$\mathbf{W}_{22}=\mathbf{0}$	trivial	almost complete MDS	complete MDS
	$\mathbf{W}_{22}\neq\mathbf{0}$	complete MDS	complete MDS	complete MDS

An interesting application appears for $\mathbf{W}_{11} = \mathbf{0}$ and $\mathbf{W}_{22} = \mathbf{0}$ where both $\mathbf{X}_1$ and $\mathbf{X}_2$ are updated. This type of analysis is called *unfolding* and its connection with MDS is elaborated in Heiser (1981). In unfolding the only interest is to find the optimal distances between the points of the two groups, irrespective of the distances within each group. There is a vast literature on (probabilistic) unfolding models that we do not discuss here. We merely wanted to note the connection with incomplete MDS.

Some nice simplifications occur in the computation of the update for unfolding. The matrix $\mathbf{V}_{11}$ simplifies to a diagonal matrix with elements $\mathbf{V}_{12}\mathbf{1}$ and similarly $\mathbf{V}_{22}$ to a diagonal matrix with elements $\mathbf{1}'\mathbf{V}_{12}$. A straightforward algorithm minimizing STRESS for unfolding is obtained by alternatingly update $\mathbf{X}_1$ while $\mathbf{X}_2$ is kept fixed and subsequently update $\mathbf{X}_2$ while $\mathbf{X}_1$ is kept fixed. The updates can be derived from (5.19). If we are prepared to compute the Moore-Penrose inverse of $\mathbf{V}$ then $\mathbf{X}_1$ and $\mathbf{X}_2$ can be updated simultaneously, which turns out to be simple when $\mathbf{W}_{12} = \mathbf{1}\mathbf{1}'$ (see Heiser, 1981, p. 177). Remember that for unidimensional unfolding we have to refer to combinatorial optimization programs (see section 1.5) to get adequate minima. If $\mathbf{X}_2$ is kept fixed all the

time, then unfolding reduces to *external unfolding* (see Heiser, 1987). Here, additional points have to be fitted into a fixed configuration that may have come from a previous analysis. Note that for external unfolding it does not matter whether only one object is updated or all the objects in $\mathbf{X}_1$ simultaneously.

Another model is obtained if in addition to external unfolding we include information on the distances and dissimilarities within group 1. We call this new model *semi-complete* scaling, i.e., $\mathbf{W}_{11} \neq \mathbf{0}$ and $\mathbf{X}_2$ is fixed. Semi-complete MDS is useful whenever we know the coordinates of a set of objects given from another analysis (e.g., a previous MDS analysis), the dissimilarities of these objects with a set of new objects, and dissimilarities between the new objects themselves. Semi-complete MDS fits the new objects optimally with respect to themselves, and the objects for which the coordinates are known. This type of analysis is therefore particularly useful when different objects are collected more than once. Another application of semi-complete MDS is moving frame analysis, which is discussed in the next section.

If the coordinates of $\mathbf{X}_2$ are free, then semi-complete MDS turns into complete MDS if $\mathbf{W}_{22} \neq \mathbf{0}$ and into *almost complete* MDS if $\mathbf{W}_{22} = \mathbf{0}$. The within group 1 distances and the between group 1 and 2 distances are fitted in almost complete MDS, over both the coordinates of group 1 and group 2. Thus the only difference between almost complete MDS and complete MDS is that the distances within group 2 are *not* fitted.

Some overlap between these models occurs in the special cases that $n_1 = 1$ and $n_1 = n - 1$. If $n_1 = 1$ and $\mathbf{X}_2$ is fixed then both external unfolding and semi-complete scaling only fit one point with respect to all the others. This could be used to for minimizing STRESS by adjusting one point at a time in a *cyclic point descent* algorithm (cf. Luenberger, 1973). However, if $\mathbf{X}_2$ is kept free and $\mathbf{W}_{22} \neq 0$ then we just have complete MDS again. For $\mathbf{X}_2$ free and $\mathbf{W}_{22} = 0$ a trivial problem with perfect fit arises, since the point of group 1 and $n - 1$ points of group 2 can always be placed on a line such that the distances between the groups match the between group dissimilarities perfectly. In fact, even if $n_1 = p$ a perfect solution can be obtained, provided $\mathbf{W}_{11} = \mathbf{0}$. This same trivial problem occurs of the role of $\mathbf{X}_1$ and $\mathbf{X}_2$ is reversed, i.e., if $n_1 = n - 1$ and $\mathbf{W}_{11} = \mathbf{0}$. However, if $\mathbf{W}_{11} \neq \mathbf{0}$ and $n_1 = n - 1$ then we deal with complete MDS if $\mathbf{X}_2$ is free and with MDS with one fixed point if $\mathbf{X}_2$ is fixed. In the latter case the columns of $\mathbf{X}$ are not necessarily centred anymore.

5.3 Moving frame multidimensional scaling

In this section we elaborate on a method proposed by Groenen and Heiser (1992) which monotonically decreases STRESS with two groups of stimuli. Iteratively a set of objects is selected by some criterion to form a frame. These objects, say in group 2, are kept fixed and the other objects are updated optimally with respect to the objects in the frame set and themselves using semi-complete MDS. Since the group of frame objects changes over the iterations, we call this method *moving frame MDS*. Our aim is to investigate whether

certain frame selection strategies can accelerate the computations when compared with complete MDS. Some numerical examples are discussed in the next section.

Other methods of accelerating the minimization of STRESS exist. For example, we could use the conjugate gradient method proposed by De Leeuw and Heiser (1980), their relaxed update (a discussion follows shortly), approximate STRESS by a quadratic function as done by Stoop and De Leeuw (1982), or use a second order minimization method, like Newton's algorithm. The last two options have the disadvantage that the property of monotone convergence of STRESS is lost. The conjugate gradient method does decrease STRESS, but needs expensive inner iterations, except if its fixed version –the relaxed update– is used. Convergence of STRESS is also retained in moving frame MDS, since it uses semi-complete MDS.

The main idea of using moving frame MDS for accelerating computations of complete MDS stems from the observation that one semi-complete scaling update requires less computational effort than one complete MDS update, especially when n_2 is large. Suppose that only a small group of points has changed substantially between two subsequent MDS updates. Then, semi-complete scaling could have been used to obtain almost the same reduction in STRESS, but would have been much faster. However, in practice we do not know in advance which points will change substantially. Therefore, the overall gain in efficiency of moving frame MDS cannot be assessed in advance. It may depend on the selection rule used, the size of n_1, the stop criterion of semi-complete MDS, and the dissimilarities that are fitted. In the next section we perform some numerical experiments to see to what extent different options in moving frame MDS results in a faster algorithm when compared to complete MDS.

The moving frame algorithm for minimizing STRESS is summarized as follows:

1. Select $r < n$ objects in group 1 to be updated, according to some selection rule. Declare all other objects to form the group of frame points (group 2).
2. Perform semi-complete scaling to find optimal updates of the group 1 objects, keeping the frame points fixed.
3. Stop if convergence is reached, otherwise go to 1.

Several rules can be used to select the frame points, three of which we discuss below. But we start by discussing some aspects of semi-complete MDS.

The update in semi-complete scaling can be determined in one step by setting the gradient of $\hat{\sigma}^2(\mathbf{X}_1,\mathbf{Y})$ in (5.19) equal to zero for $\mathbf{X}_2 = \mathbf{Y}_2$. This gives

$$\begin{aligned} 2\mathbf{V}_{11}\mathbf{X}_1 - 2\mathbf{B}_{11}(\mathbf{Y})\mathbf{Y}_1 + 2\mathbf{V}_{12}\mathbf{X}_2 - 2\mathbf{B}_{12}(\mathbf{Y})\mathbf{Y}_2 &= 0 \\ \overline{\mathbf{X}}_1 = \mathbf{V}_{11}^{-1}\big(\mathbf{B}_{11}(\mathbf{Y})\mathbf{Y}_1 - \mathbf{V}_{12}\mathbf{Y}_2 + \mathbf{B}_{12}(\mathbf{Y})\mathbf{Y}_2\big), & \end{aligned} \tag{5.20}$$

which is the update formula needed.

The semi-complete scaling algorithm exhibits one defect over the complete MDS algorithm. The SMACOF algorithm is insensitive to normalization of the dissimilarities,

because in each step of the algorithm an optimal normalization of $\mathbf{X}$ is computed automatically. In semi-complete MDS $\mathbf{X}_2$ is fixed, so that optimal normalization is not possible. We could loosen this constraint by requiring that $\mathbf{X}_2 = \beta\mathbf{Y}_2$. Suppose $\overline{\mathbf{X}}_1$ is obtained with (5.20). Then we search for a β that minimizes $\sigma^2(\beta\overline{\mathbf{X}}_1, \beta\mathbf{X}_2)$ for $\overline{\mathbf{X}}_1$ and $\mathbf{X}_2$ fixed. Since $d_{ij}(\mathbf{X})$ is a positive homogenous function, we have

$$\sigma^2(\beta\mathbf{X}) = \eta_\delta^2 + \beta^2\eta^2(\mathbf{X}) - 2\beta\rho(\mathbf{X}) \tag{5.21}$$

which has a minimum of $\eta_\delta^2 - \rho^2(\mathbf{X})/\eta^2(\mathbf{X})$ for $\beta = \rho(\mathbf{X})/\eta^2(\mathbf{X})$ (see De Leeuw, 1977). If β deviates from 1, STRESS is reduced even more, without much computational effort. For β close to 1, the effort may be too large compared to the decrease in STRESS. Let $\beta = 1 - \alpha$ for some small $|\alpha| < 0.001$. Then, (5.21) changes into

$$\begin{aligned}\sigma^2(\beta\mathbf{X}) &= \eta_\delta^2 + \eta^2(\mathbf{X}) - 2\rho(\mathbf{X}) + (\alpha^2 - 2\alpha)\eta^2(\mathbf{X}) + 2\alpha\rho(\mathbf{X}) \\ &= \sigma^2(\mathbf{X}) + \alpha(\alpha - 2)\eta^2(\mathbf{X}) + 2\alpha\rho(\mathbf{X}),\end{aligned} \tag{5.22}$$

which shows that only a marginal decrease of STRESS is obtained. Therefore, we shall use this only when $|\beta - 1| > 0.001$.

The efficiency of moving frame MDS is partly determined by the stopping criterion of semi-complete MDS. If the stopping criterion is fixed and too loose, only one update is obtained for each selection of group 1. If the selection procedure is computationally intensive much of the advantage may be lost. A similar problem occurs when the convergence criterion (difference in subsequent STRESS values) of semi-complete MDS is larger than the one of the moving frame scaling: then too only one semi-complete update is taken. Therefore, we set the convergence criterion of semi-complete scaling to be 0.1 times the average reduction in STRESS by the last five iterations of moving frame MDS.

Without trying to be complete we discuss three rational decision rules for selecting frame objects. The first decision rule is based on the accumulation of badly fitting points. For a given configuration $\mathbf{X}$ the STRESS accounted for by object i is given by

$$\tfrac{1}{2}\textstyle\sum_{j=1}^{n} w_{ij}(\delta_{ij} - d_{ij}(\mathbf{X}))^2 \,. \tag{5.23}$$

The first decision rule selects points that contribute least to STRESS in the frame set, since we do not expect to gain much by updating points that have a good fit. The second and third decision rule are based on the (sub-)gradient of the configuration. A sharp decrease of STRESS is expected of points that have a subgradient deviant from zero. The subgradient of STRESS, presented by

$$\nabla\sigma^2(\mathbf{X}) = 2n\mathbf{V}\mathbf{X} - 2\,\mathbf{B}(\mathbf{X})\mathbf{X}, \tag{5.24}$$

equals the gradient whenever there are no two points i and j that have $d_{ij}(\mathbf{X}) = 0$. The second and third decision rule amount to choosing the n_2 points that have the lowest sum

of the absolute and the squared row values of $\nabla\sigma^2(\mathbf{X})$, respectively. Taking the sum of squared coordinates for each point emphasizes larger values of the gradient. The gradient based decision rules exclude points that are expected to change little by being updated.

Finally, the majorization results remain valid if we use the relaxed update of De Leeuw and Heiser (1980). They showed that the update $\mathbf{X}^+ \leftarrow \overline{\mathbf{X}} + \alpha(\mathbf{Y} - \overline{\mathbf{X}})$,with $\overline{\mathbf{X}}$ the Guttman transform and $\mathbf{Y}$ the previous configuration, yields a decreasing series of STRESS values for $-1 < \alpha < 1$. For $\alpha = -1$, the relaxed update, the number of iterations was found to be approximately halved.

In the next section we discuss the effect of the options of moving frame MDS on accelerating the computations for finding the minimum of STRESS.

5.3.1 Numerical experiments with moving frame MDS

Here, we report numerical experiments with moving frame MDS to see if it can accelerate the computations of complete MDS. We compared several options on two datasets, the Andrew's cross taken from Spence (1982) with $n = 40$ and a distance matrix derived from the four independent variables of the Iris data of Fisher (1936) with $n = 150$. We compared several options: complete MDS, cyclic point descent and moving frame MDS with three different selection rules. Moreover, all the runs were computed twice, one time with and one time without the relaxed update. For each option, the STRESS value is plotted against the CPU time. Although we did additional experiments, they are left out here, because the results did not differ from those reported here.

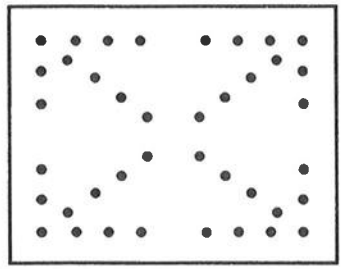


Figure 5.4 *The points of the Andrew's cross used by Spence (1982).*

The Andrew's cross consists of 40 points arranged as in Figure 5.4. The distances between the points were used as dissimilarities, so that the global minimum has zero STRESS. In Figure 5.5 the results of the analysis is given, where for convenience complete MDS is reported in both plots. It turns out that complete MDS is the fastest method for reducing STRESS. A good alternative is the cyclic point descent algorithm and the maximum STRESS decision rule with moving frame MDS. The gradient based decision

rules perform badly, using much more CPU than complete MDS to reach the same minimum.

The analysis was repeated using the relaxed update. In Figure 5.6b we see that complete MDS using the relaxed update stops too early with a very large STRESS. This effect can be ascribed to the incapability of the relaxed update to scale the configuration to the appropriate size, in contrast to the unrelaxed complete MDS update. Adjusting the scale size for complete MDS analysis with the relaxed update shows fast convergence indeed.

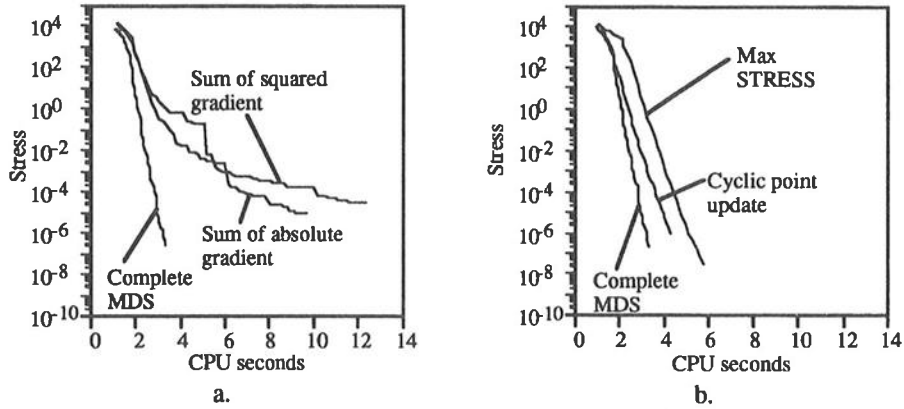


Figure 5.5 *Results of moving frame MDS for example the Andrew's cross, 40 objects. For convenience the results of complete MDS is given in both plots*

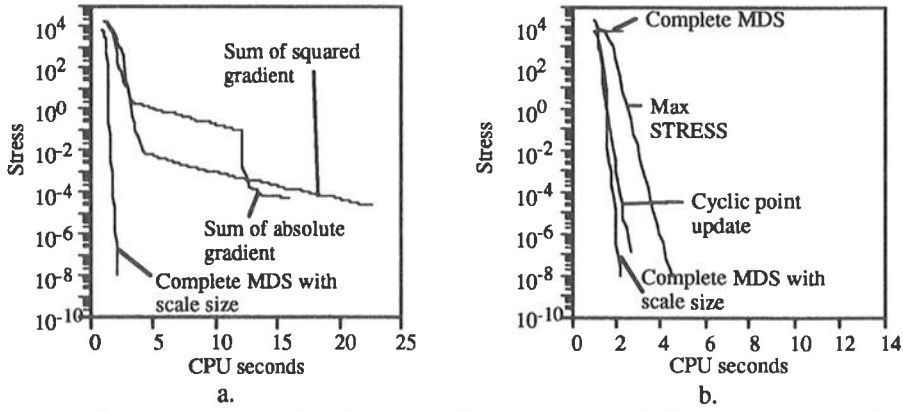


Figure 5.6 *Results of moving frame MDS for example the Andrew's cross, 40 objects, using the relaxed update.*

Therefore, we used the scale size adjustment whenever the relaxed update was used (for computational details, see the previous section). From this experiment we may conclude that moving frame MDS using the gradient based selection rules does not accelerate the computations. The maximum STRESS decision rule and the cyclic point update behave better, but not as good as complete MDS. The fastest method reaching the minimum is using complete MDS with the relaxed update where the configuration is scaled to the appropriate size.

The second dataset, distances based on the 150 objects of Fisher's Iris data, was used to see how moving frame MDS performs for a dataset with a large number of objects and a non-zero STRESS at the minimum. The analysis was done in two dimension, while the objects vary in four dimensions. In Figure 5.7, the results are presented for the unrelaxed update. We excluded the gradient based selection rules for moving frame MDS, because of their bad performance in the Andrew's cross example. In Figure 5.7a we have plotted the difference of STRESS and the lowest STRESS value found against CPU time. In Figure 5.7b we plotted the difference in STRESS between two subsequent iterations against CPU, to show the acceleration and deceleration of the reduction in STRESS. We see that the maximum STRESS decision rule with moving frame MDS stops too early with a very high STRESS and again that complete MDS performs better than cyclic point update. It seems that cyclic point descent algorithm follows the complete MDS algorithm at a somewhat slower pace. Redoing the analysis using the relaxed update, reported in Figure 5.8, shows that the CPU time is approximately halved for all options.

From these experiments we may conclude that the moving frame MDS does *not* accelerate computations for minimizing STRESS. A possible cause could be that moving frame MDS does not provide enough freedom to model the relations between all

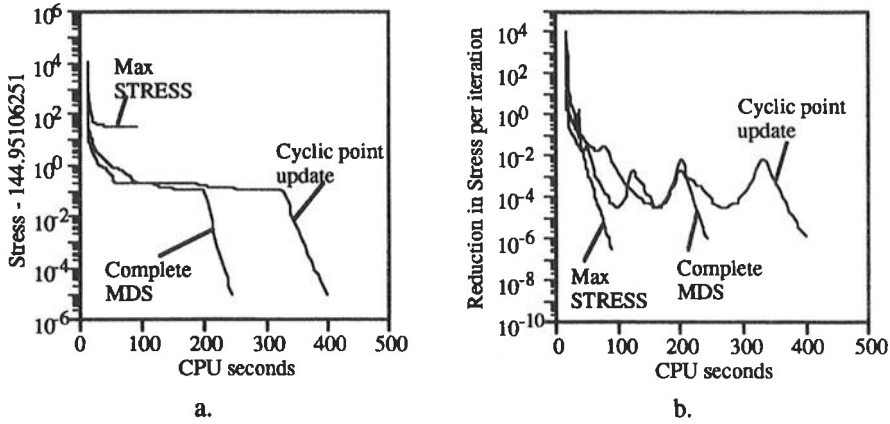


Figure 5.7 *Results of moving frame MDS for example of 150 objects.*

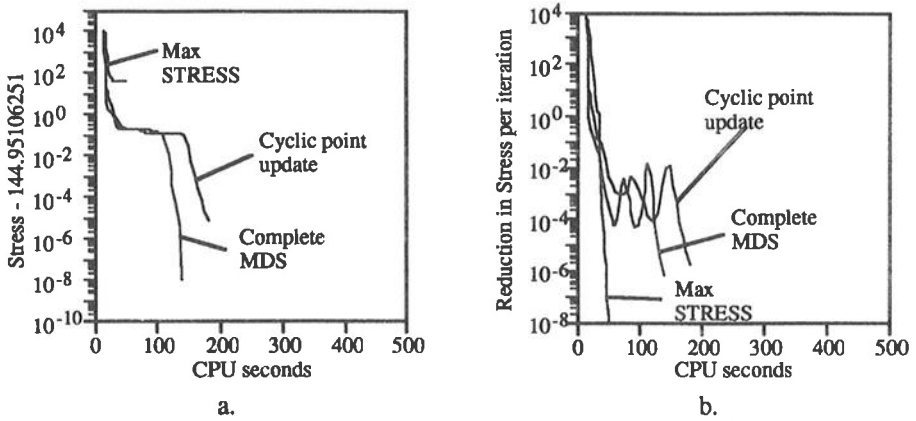


Figure 5.8 *Results of moving frame MDS for example of 150 objects with the relaxed update.*

objects at the same time. Other reasons for failure could be that the selection rules take too much time or that we simply need better selection rules.

Complete MDS is the fastest strategy available in this study. Furthermore, the relaxed update may halve the CPU time. However, when using the relaxed update it is of great importance to scale the configuration matrix properly, otherwise the algorithm may stop at high STRESS values.

5.4 Discussion and conclusions

In this chapter we discussed several computational aspects of incomplete MDS. First, we treated three structured designs –partitioned block design, the block tridiagonal design, and the circular design– and showed how the computation of the inverse needed for these designs can be accelerated dramatically. Then, we discussed a classification of models based on splitting the set of objects into two parts. Using this classification some known models (like (external) unfolding and complete MDS) and new models (cyclic point descent, semi-complete MDS, almost complete MDS) were presented. Finally, we discussed an application of semi-complete MDS aimed at accelerating the MDS algorithm. We called this moving frame MDS, since we iteratively select a frame of well fitting points, keep them fixed, and fit the other points using semi-complete MDS. Numerical experiments showed that moving frame MDS was always much slower than complete MDS. The fastest computations were reached for complete MDS using the relaxed update. It turns out that when using the relaxed update we always have to adjust the scaling size of the coordinates. Otherwise, the complete MDS algorithm may stop too early.

CHAPTER 6

CLUSTER DIFFERENCES SCALING

Several problems exist with MDS of large datasets. In the previous chapter we discussed some computational aspects of accelerating computations for some structured designs that can be used to reduce the number of object pairs to be considered. In this chapter, we focus on some interpretational difficulties that can arise when having a large number of objects. Interpretation can become difficult if we have to inspect too many points at the same time. One solution to this problem is to perform a cluster analysis on the coordinates of the objects after an MDS analysis. However, this strategy ignores the information on how well the distances fit the dissimilarities between the objects. Therefore, we elaborate on a method proposed by Heiser (1992) which is called *cluster differences scaling* (CDS) that performs simultaneously scaling of the clusters and clustering of the objects such that the distances fit the dissimilarities as close as possible. We propose a decomposition of STRESS into within and between cluster components. We also show that CDS can be seen as MDS with cluster restrictions on the configuration. Furthermore, we propose *fuzzy* cluster differences scaling, in which an object can be assigned to more than one cluster. We also discuss the use of repeated fuzzy CDS to diminish the problem of local minima that may occur with CDS.

6.1 Clustering with scaling of the clusters

The basic feature of CDS is that clusters of objects are scaled, not the objects themselves. This may be done to reduce the number of different object points in the configuration, or to provide a classification of similar objects. Objects are assigned to one of K clusters, which are represented in a p-dimensional space by the $K \times p$ coordinate matrix $\mathbf{X}$ in such a way that STRESS is minimized. Let $g_{ik} = 1$ if object i belongs to cluster k and $g_{ik} = 0$ otherwise, so that the indicator matrix $\mathbf{G}$ contains the cluster memberships. Note that each object i belongs to only one of the clusters. Then STRESS can be written as

$$\sigma^2(\mathbf{G},\mathbf{X}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K}\sum_{l=1}^{K} g_{ik}g_{jl}w_{ij}\left(\delta_{ij} - d_{kl}(\mathbf{X})\right)^2 \qquad (6.1)$$

where δ_{ij} is the dissimilarity between object i and j, $d_{kl}(\mathbf{X})$ is the Euclidean distance between cluster point k and cluster point l. The sum of squares given by (6.1) can be decomposed in an ANOVA-like way into four parts, by making use of the weighted Sokal-Michener distance $\tilde{\delta}_{kl} = (\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ik}g_{jl}w_{ij})^{-1} \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ik}g_{jl}w_{ij}\delta_{ij}$. The decomposition is given by

$$
\begin{aligned}
\sigma^2(\mathbf{X}) = {} & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K}\sum_{\substack{l=1 \\ l\neq k}}^{K} g_{ik}g_{jl}w_{ij}\left(\delta_{ij} - \tilde{\delta}_{kl}\right)^2 + && \text{between cluster object STRESS} \\
& \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K} g_{ik}g_{jk}w_{ij}\left(\delta_{ij} - \tilde{\delta}_{kk}\right)^2 + && \text{within cluster object STRESS} \\
& \sum_{k=1}^{K}\sum_{\substack{l=1 \\ l\neq k}}^{K}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ik}g_{jl}w_{ij}\right)\left(\tilde{\delta}_{kl} - d_{kl}(\mathbf{X})\right)^2 + && \text{between cluster STRESS} \\
& \sum_{k=1}^{K}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ik}g_{jk}w_{ij}\right)\tilde{\delta}_{kk}^2 && \text{within cluster STRESS.} \qquad (6.2)
\end{aligned}
$$

The first term gives the STRESS due to the differences between the object dissimilarities and the average cluster dissimilarities. The second term shows how much STRESS can be ascribed to the difference between an object pair and the average dissimilarity of the cluster it belongs to. The within cluster-object STRESS measures the deviance of the dissimilarities of the objects inside a cluster with the average cluster dissimilarity. The third term gives the between cluster STRESS. It merely shows how well the distances between the cluster points look like the intercluster dissimilarity. Since the points in one cluster necessarily have zero distance but do not have (necessarily) zero dissimilarities, a final term has to be included giving the STRESS within a cluster, due to the dissimilarities being nonzero within a cluster. The within cluster STRESS measures the deviance from zero of the Sokal-Michener distance of a cluster.

Heiser (1992) discusses the case in which the second and fourth term of (6.2), the within cluster terms, are excluded by definition. The main reason for doing so, is to avoid ball shaped clusters that are located in the centroid of their cluster members. In this manner Heiser allows for other cluster shapes, like elongated forms or sausage shapes. As a side effect the STRESS values are generally lower, because no within cluster STRESS is included. Here, we include the within cluster terms so that the direct relation between ordinary MDS and CDS is preserved.

The decomposition (6.2) implies that STRESS can be minimized in two steps: an adjusted multidimensional scaling step and a clustering step. This outer algorithm is solved in an alternating fashion. First, a (sub)optimal configuration $\mathbf{X}$ is found for fixed clusters $\mathbf{G}$. Then, we fix $\mathbf{X}$ and search for (sub)optimal clusters $\mathbf{G}$. We return to the first step until convergence is reached.

The only term of (6.2) dependent on $\mathbf{X}$ is the between cluster STRESS. Clearly, for fixed $\mathbf{G}$ this term is equivalent to a MDS problem with special weights, which may be minimized by the ordinary SMACOF algorithm. The between cluster STRESS can be expressed as

$$\sigma^2_{BC}(\mathbf{X}) = \sum_k\sum_l(\sum_i\sum_j g_{ik}g_{jl}w_{ij})\tilde{\delta}^2_{kl} + \text{tr}\mathbf{X}'\mathbf{V}\mathbf{X} - 2\text{tr}\mathbf{X}'\mathbf{B}(\mathbf{X})\mathbf{X}, \tag{6.3}$$

where the matrix $\mathbf{B}(\mathbf{X})$ has off diagonal elements $b_{kl} = -(\sum_i\sum_j g_{ik}g_{jl}w_{ij})\tilde{\delta}_{kl}/d_{kl}(\mathbf{X})^2$ and diagonal elements $b_{kk} = -\sum_l b_{kl}$, and $\mathbf{V}$ has off diagonal elements $v_{kl} = -\sum_i\sum_j g_{ik}g_{jl}w_{ij}$ and diagonal elements $v_{kk} = -\sum_l v_{kl}$. Conforming to the SMACOF theory, the update $\mathbf{X}^+$, given by the Guttman transform, is

$$\mathbf{X}^+ = \mathbf{V}^-\mathbf{B}(\mathbf{X})\mathbf{X} \tag{6.4}$$

where $\mathbf{V}^-$ is the Moore-Penrose inverse of $\mathbf{V}$. If each cluster contains at least one point, $\mathbf{V}$ has rank K–1 because the rows and columns add to zero.

Some nice simplifications occur if all weights w_{ij} are equal to one. First, the unweighted Sokal-Michener distance can be expressed as $\tilde{\delta}_{kl} = (n_k n_l)^{-1}\sum_{i=1}^n\sum_{j=1}^n g_{ik}g_{jl}\delta_{ij}$, where n_k is the number of objects in cluster k. Furthermore, the weight matrix can be expressed as

$$\mathbf{V} = \text{diag}(\mathbf{nn}'\mathbf{1}) - \mathbf{nn}', \tag{6.5}$$

where $\mathbf{n}$ is the vector with elements n_k, the number of objects per cluster. With some algebra a significant simplification of the computation of the inverse is obtained. In fact, it suffices to compute $\mathbf{S}^{-1}$, where $\mathbf{S}$ denotes the diagonal matrix diag($\mathbf{nn}'\mathbf{1}$). We have to show that the Moore-Penrose inverse of $\mathbf{V}$ equals $\mathbf{V}^- = \mathbf{J}\mathbf{S}^{-1}\mathbf{J}$, with $\mathbf{J}$ the centering operator $\mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}'$, and that the usual conditions for a Moore-Penrose inverse apply. We first show that

$$(\mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}')\mathbf{S}^{-1}(\mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}')\mathbf{V} = \mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}'. \tag{6.6}$$

Using $\mathbf{S}\mathbf{1} = (\mathbf{1}'\mathbf{n})\mathbf{n}$ and $\mathbf{S}^{-1}\mathbf{n} = (\mathbf{1}'\mathbf{n})^{-1}\mathbf{1}$ we obtain

$$\begin{aligned}(\mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}')\mathbf{S}^{-1}(\mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}')(\mathbf{S} - \mathbf{nn}') = (\mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}')\mathbf{S}^{-1}(\mathbf{S} - \mathbf{nn}') = \\ (\mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}')(\mathbf{I} - \frac{\mathbf{1}\mathbf{n}'}{\mathbf{1}'\mathbf{n}}) = \mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}' - \frac{\mathbf{1}\mathbf{n}'}{\mathbf{1}'\mathbf{n}} + \frac{\mathbf{1}\mathbf{n}'}{\mathbf{1}'\mathbf{n}} = \mathbf{I} - K^{-1}\mathbf{1}\mathbf{1}',\end{aligned} \tag{6.7}$$

which proves $\mathbf{V}^-\mathbf{V} = \mathbf{J}$. Similarly, $\mathbf{V}\mathbf{V}^- = \mathbf{J}$. This proves that $\mathbf{V}^-\mathbf{V}$ and $\mathbf{V}\mathbf{V}^-$ are symmetric. Since $\mathbf{J}\mathbf{V}^- = \mathbf{V}^-\mathbf{J} = \mathbf{V}^-$ and $\mathbf{J}\mathbf{V} = \mathbf{V}\mathbf{J} = \mathbf{V}$, we have $\mathbf{V}\mathbf{V}^-\mathbf{V} = \mathbf{V}$ and $\mathbf{V}^-\mathbf{V}\mathbf{V}^- = \mathbf{V}^-$ so that $\mathbf{J}\mathbf{S}^{-1}\mathbf{J}$ satisfies the conditions of the Moore-Penrose inverse of $\mathbf{V}$. Since $\mathbf{S}^{-1}$ is multiplied by $\mathbf{B}(\mathbf{X})\mathbf{X}$ which is column centered, the update formula becomes $\mathbf{X}^+ = \mathbf{J}\mathbf{S}^{-1}\mathbf{B}(\mathbf{X})\mathbf{X}$ in case all weights w_{ij} are unity. Applying this formula greatly accelerates the

computation of the inverse, especially when the number of clusters K is large. The effect can be significant, since $\mathbf{V}^-$ has to be computed whenever a change in the cluster membership occurred in the other step of the outer algorithm.

The second step of the algorithm is to form optimal clusters. Let us focus on the cluster membership of object i only, keeping the cluster memberships of the other objects fixed. By writing STRESS as

$$\sigma^2(\mathbf{G},\mathbf{X}) = \sum_i\sum_k g_{ik}\sum_j\sum_l g_{jl}w_{ij}(\delta_{ij} - d_{kl}(\mathbf{X}))^2 = \sum_i\sum_k g_{ik}\gamma_{ik} \tag{6.8}$$

it can be seen that the largest reduction in STRESS is obtained if object i is assigned to cluster k with $\gamma_{ik} = \min_l\gamma_{il}$. This may be done for one object at a time, all the objects after each other or more iterations over all the objects until no change occurs. We return to these strategies shortly. If no extra constraints are set, clusters without objects may appear. Clearly, this is not desirable as may be seen from (6.2) which shows that STRESS will occur whenever two or more objects points belong to one cluster. Therefore, we require that every cluster contains at least one object and has $w_{ij}\delta_{ij} > 0$. The current allocation procedure can be seen as a special application of the K-means algorithm of MacQueen (1967); a loss function is minimized by assigning an object to the cluster that gives the largest decrease in the loss function.

Since the clustering step is only one of the two steps in our alternating least squares algorithm, several strategies are possible to stop the clustering step and continue with the scaling step. The first strategy is to stop the clustering step if object i changes from one cluster to another and continue with the MDS step. In the next clustering step we continue cycling through the objects starting with object i+1. The second strategy is to cycle through all objects once and reallocate them to the cluster that yields the lowest STRESS. The third strategy is a simple extension of the second: continue the cycling until no reallocation occurs. All three strategies should meet the restriction that each cluster contains at least one object. Therefore, we do not change an object if it is the last one in a cluster. Whatever strategy is used, we always have to base the computations on the most recent partitioning in clusters. Thus, if object i is changed from cluster k to l, then the computations for object i+1 are based on object i being in cluster l. With this convention we can guarantee that every reallocation reduces STRESS.

The amount of reduction in STRESS may vary between the two steps. For example, if the clustering step yields no reallocation, STRESS does not change either. During the first few steps it is expected that the clustering steps decrease STRESS drastically. It seems therefore wise to have a rather weak convergence criterion for the scaling step if large drops in STRESS occur in the cluster step. The convergence criterion may be set more tightly if the clustering step results in only small reductions. To smooth this process we set the convergence criterion of the scaling step in our examples to 10^{-3} times the reduction in the previous MDS step and clustering step. This prevents us from wasting too much time in the MDS step at the beginning, while accuracy is obtained near the end.

6.1.1 Approaching CDS as MDS with cluster restrictions on the configuration

An alternative way of looking at CDS is to regard it as MDS with clustering restrictions on the configuration. Then, minimizing CDS is viewed as minimization of the ordinary STRESS function where the configuration is restricted to be of the form $\mathbf{GX}$. To see this consider the following. Expression (6.1) of $\sigma^2(\mathbf{G},\mathbf{X})$ can be rewritten as

$$\begin{aligned}\sigma^2(\mathbf{G},\mathbf{X}) &= \sum_i\sum_j\sum_k\sum_l g_{ik}g_{jl}w_{ij}(\delta_{ij} - d_{kl}(\mathbf{X}))^2 \\ &= \sum_i\sum_j w_{ij}(\sum_k\sum_l g_{ik}g_{jl}\delta_{ij} - \sum_k\sum_l g_{ik}g_{jl}d_{kl}(\mathbf{X}))^2 \\ &= \sum_i\sum_j w_{ij}(\delta_{ij} - \sum_k\sum_l g_{ik}g_{jl}d_{kl}(\mathbf{X}))^2 \qquad (6.9)\end{aligned}$$

using the fact that $\sum_k\sum_l g_{ik}g_{jl} = 1$. Also, we have

$$\begin{aligned}\sum_k\sum_l g_{ik}g_{jl}d_{kl}(\mathbf{X}) &= \sum_k\sum_l g_{ik}g_{jl}\|\mathbf{x}_k - \mathbf{x}_l\| = \|(\sum_k\sum_l g_{ik}g_{jl}\mathbf{x}_k) - (\sum_k\sum_l g_{ik}g_{jl}\mathbf{x}_l)\| \\ &= \|(\sum_k g_{ik}\mathbf{x}_k\sum_l g_{jl}) - (\sum_l g_{jl}\mathbf{x}_l\sum_k g_{ik})\| = \|(\sum_k g_{ik}\mathbf{x}_k) - (\sum_l g_{jl}\mathbf{x}_l)\| \\ &= \|\mathbf{g}_i\mathbf{X} - \mathbf{g}_j\mathbf{X}\| = d_{ij}(\mathbf{GX}), \qquad (6.10)\end{aligned}$$

because $\sum_k g_{jk} = 1$. Using (6.9) and (6.10) allows us to write

$$\sigma^2(\mathbf{G},\mathbf{X}) = \sum_i\sum_j w_{ij}(\delta_{ij} - \sum_k\sum_l g_{ik}g_{jl}d_{kl}(\mathbf{X}))^2 = \sum_i\sum_j w_{ij}(\delta_{ij} - d_{ij}(\mathbf{GX}))^2 \qquad (6.11)$$

so that $\sigma^2(\mathbf{G},\mathbf{X}) = \sigma^2(\mathbf{GX})$. Here, we see that CDS can be seen as ordinary MDS with cluster restrictions on the configuration. This formulation allows us to express CDS as

$$\sigma^2(\mathbf{GX}) = \eta_\delta^2 + \mathrm{tr}\,\mathbf{X'G'VGX} - 2\,\mathrm{tr}\mathbf{X'G'B(GX)GX}, \qquad (6.12)$$

where as usual $\mathbf{V}$ is an $n\times n$ matrix with off-diagonal elements $v_{ij} = -w_{ij}$ and diagonal elements $v_{ij} = -\sum_j v_{ij}$. Note that many zero distances occur because of the cluster restrictions. As shown in 1.3 the STRESS function can be majorized by

$$\sigma^2(\mathbf{GX}) \le \hat{\sigma}^2(\mathbf{GX},\mathbf{FY}) = \eta_\delta^2 + \mathrm{tr}\,\mathbf{X'G'VGX} - 2\,\mathrm{tr}\mathbf{X'G'B(FY)FY}, \qquad (6.13)$$

where $\mathbf{F}$ is an indicator matrix giving the cluster memberships of the previous iteration, and $\mathbf{Y}$ is coordinate matrix from the previous iteration. Then, the unrestricted Guttman transform is given by

$$\overline{\mathbf{X}} = \mathbf{V}^-\mathbf{B(FY)FY}. \qquad (6.14)$$

We only have to find a $\mathbf{G}^+$ and $\mathbf{X}^+$ such that

$$\|\mathbf{G}^+\mathbf{X}^+ - \overline{\mathbf{X}}\|_{\mathbf{V}}^2 \le \|\mathbf{FY} - \overline{\mathbf{X}}\|_{\mathbf{V}}^2. \qquad (6.15)$$

Such an $\mathbf{G}^+\mathbf{X}^+$ can be found by weighted K-means clustering on the configuration $\overline{\mathbf{X}}$. The weighted K-means algorithm is a slight adjustment of the ordinary K-means algorithm, which can be computed as follows. We start as initial values for $\mathbf{G}$ and $\mathbf{X}$, with $\mathbf{F}$ and $\mathbf{Y}$. Then $\|\mathbf{GX} - \overline{\mathbf{X}}\|_{\mathbf{V}}^2$ is minimized alternatingly over $\mathbf{X}$ for fixed $\mathbf{G}$, and over $\mathbf{G}$ for fixed $\mathbf{X}$. The conditional minimum of $\|\mathbf{GX} - \overline{\mathbf{X}}\|_{\mathbf{V}}^2$ over $\mathbf{X}$ for fixed $\mathbf{G}$ is given by $(\mathbf{G}'\mathbf{V}\mathbf{G})^-\mathbf{G}'\mathbf{V}\overline{\mathbf{X}}$. How the reallocation should be done is less simple to see. The minimization problem can be written as

$$\begin{aligned}\|\mathbf{GX} - \overline{\mathbf{X}}\|_{\mathbf{V}}^2 &= \text{tr}\,(\mathbf{GX} - \overline{\mathbf{X}})'\mathbf{V}(\mathbf{GX} - \overline{\mathbf{X}}) \\ &= \text{tr}\,\overline{\mathbf{X}}'\mathbf{V}\overline{\mathbf{X}} + \text{tr}\,\mathbf{X}'\mathbf{G}'\mathbf{V}\mathbf{G}\mathbf{X} - 2\,\text{tr}\,\mathbf{X}'\mathbf{G}'\mathbf{V}\overline{\mathbf{X}} \\ &= \text{tr}\,\overline{\mathbf{X}}'\mathbf{V}\overline{\mathbf{X}} + \text{tr}\,\mathbf{G}'\mathbf{V}\mathbf{G}\mathbf{X}\mathbf{X}' - 2\,\text{tr}\,\mathbf{V}\overline{\mathbf{X}}\mathbf{X}'\mathbf{G}'. \qquad (6.16)\end{aligned}$$

The vec and Kronecker product notation (see e.g., Magnus and Neudecker, 1988) allows us to write $\text{tr}\,\mathbf{G}'\mathbf{V}\mathbf{G}\mathbf{X}\mathbf{X}' = \text{vec}(\mathbf{G}')'(\mathbf{V}\otimes\mathbf{X}\mathbf{X}')\text{vec}(\mathbf{G}')$. Some rewriting shows that

$$\text{tr}\,\mathbf{G}'\mathbf{V}\mathbf{G}\mathbf{X}\mathbf{X}' = \sum_{i=1}^n\sum_{j=1}^n\sum_{k=1}^K\sum_{l=1}^K v_{ij}g_{ik}g_{jl}\mathbf{x}_k'\mathbf{x}_l, \qquad (6.17)$$

where $\mathbf{x}_k$ is the kth row of $\mathbf{X}$. Furthermore, let $\mathbf{A} = \mathbf{V}\overline{\mathbf{X}}\mathbf{X}'$, then $\text{tr}\,\mathbf{V}\overline{\mathbf{X}}\mathbf{X}'\mathbf{G}' = \text{tr}\,\mathbf{A}\mathbf{G}' = \sum_{i=1}^n\sum_{k=1}^K g_{ik}a_{ik}$. Now we can write for (6.16)

$$\begin{aligned}\|\mathbf{GX} - \overline{\mathbf{X}}\|_{\mathbf{V}}^2 &= \text{tr}\,\overline{\mathbf{X}}'\mathbf{V}\overline{\mathbf{X}} + \sum_{i=1}^n\sum_{j=1}^n\sum_{k=1}^K\sum_{l=1}^K v_{ij}g_{ik}g_{jl}\mathbf{x}_k'\mathbf{x}_l - 2\sum_{i=1}^n\sum_{k=1}^K g_{ik}a_{ik} \\ &= \text{tr}\,\overline{\mathbf{X}}'\mathbf{V}\overline{\mathbf{X}} + \sum_{i=1}^n\sum_{k=1}^K g_{ik}\Big(\Big(\sum_{j=1}^n\sum_{l=1}^K v_{ij}g_{jl}\mathbf{x}_k'\mathbf{x}_l\Big) - 2a_{ik}\Big) \\ &= \text{tr}\,\overline{\mathbf{X}}'\mathbf{V}\overline{\mathbf{X}} + \sum_{i=1}^n\sum_{k=1}^K g_{ik}\gamma_{ik}. \qquad (6.18)\end{aligned}$$

Minimizing (6.16) over g_{ik} keeping $\mathbf{X}$ and g_{jl} fixed for $j \neq i$ is obtained by setting $g_{ik}^+ = 1$ if $\gamma_{ik} = \min_l \gamma_{il}$ and $g_{ik}^+ = 0$ otherwise. In case all weights are equal to one, the allocation procedure for finding optimal cluster memberships amounts to ordinary K-means clustering. Once a reallocation of object i to a different cluster occurred, we could update $\mathbf{X}$ again. Alternatively, we might also continue updating the other cluster memberships, and then return to updating the $\mathbf{X}$ for fixed cluster memberships. We could either be satisfied with one iteration, or could iterate until convergence of the weighted K-means algorithm occurs. Either way, inequality (6.15) holds.

The minimization algorithm outlined in this section differs from the one of the previous section. Here, we try to find a new configuration that is restricted to be of the form $\mathbf{GX}$. In the previous section we operated directly on the STRESS function and found the update for $\mathbf{G}$ and $\mathbf{X}$ by working the constraints out directly in the STRESS formula. It can be argued that the method from the previous section remains closer to the CDS function, since the clustering step operates directly on the CDS function.

The advantage of the current formulation is that it shows the relation between CDS and MDS quite clearly. However, it does not give insight in the decomposition of CDS in within and between cluster STRESS.

6.1.2 Residual analysis

The analysis outlined in the previous section results in clusters and coordinates for these clusters. The only thing we know about the objects is their cluster membership. In some situations we would still like to have a graphical representation of all objects. With residual analysis such a representation can be obtained. Several aspects can be emphasized by using different forms of residual analysis, as is discussed below.

We could fit the objects in a cluster separately for each cluster. The coordinates of the objects in cluster k, $\mathbf{X}_k$, are constrained to have their centroid in cluster point k. In this way the objects are expected to be located quite near their cluster centre. First, the residuals of the within cluster object STRESS can be decomposed. The square of the residuals are given by

$$r_{ij}^2 = \sum_k \sum_l g_{ik} g_{jl} w_{ij} (\delta_{ij} - \tilde{\delta}_{kk})^2, \tag{6.19}$$

which are used as dissimilarities for an MDS analysis for each cluster k separately. Note that $\mathbf{X}_k$ may be rotated in any way, since no information is used to attach them to other cluster points. The rotation is fixed, if we additionally require that the Sokal-Michener distance between the points of cluster k and the other cluster points l matches their distances. This analysis can be done by using semi-complete MDS (see section 5.2). We do not expect a good fit, because the residuals (6.19) generally are not Euclidean, since some $\delta_{ij} - \tilde{\delta}_{kk}$ may be negative, others positive. Another possibility for residual analysis is to use only the Sokal-Michener distances, which may be fitted with external unfolding (see section 5.2). Or, we might compute one iteration of MDS on the dissimilarities starting from the configuration $\mathbf{GX}$ and use the resulting coordinates as supplementary points. Finally, we may use the K-means distances that were used to allocate point i to cluster l. The options are listed in Table 6.1.

Table 6.1 *Some options for fitting original points in CDS solution*

Used distances	Analysis
Within cluster residuals	MDS
Within cluster residuals and Sokal-Michener	semi-complete scaling
Sokal-Michener	external unfolding
Dissimilarities	one Guttman transform
K-means distances	external unfolding

6.1.3 Iris data

To illustrate CDS, we used the Iris data of Fisher (1936). Our objective is to see if CDS finds the appropriate clusters that agrees with their species classification. We used 25 clusters instead of three (the number of species) to mimic elongated clusters. We took the four descriptive variables in deviation from their mean and unit normalized. The Euclidean distance matrix of the 150 leaves was used as dissimilarity matrix. We generated a startconfiguration as follows. A classical scaling was performed to find the 150 coordinates in 2 dimensions. The leaves were allocated to the initial clusters by using order: leave 1 in cluster 1, leave 2 in cluster 2, ..., leave 25 in cluster 25, leave 26 in cluster 1, and so on. Each cluster point in the initial configuration was set to the centroid of classical scaling solution of the points belonging to that cluster. The convergence criterion of the outer steps of CDS was set to 10^{-6} and the convergence criterion of the weighted MDS step was set to 10^{-3} times the previous difference in outer steps. The total STRESS at the minimum is 9.176373 and its decomposition is reported in Table 6.2. The largest part of the STRESS is attributed by the between cluster object STRESS. The between cluster STRESS is relatively

Table 6.2 *Decomposition of STRESS for CDS with 25 clusters on the distance matrix derived from the four descriptive variables of the Iris data.*

Source of STRESS	STRESS	% total STRESS
Between cluster object	6.561352	71.5
Within cluster object	0.465762	5.1
Between cluster	0.788571	8.6
Within cluster	1.360688	14.8
Total	9.176373	100.0

low, which means that there is adequate fit of the cluster coordinates in Figure 6.1. Clearly, three bands that coincide with the species membership can be distinguished in the plot. Indeed, 25 clusters is enough to find elongated clusters. At the boundary of the bands we find cluster points that have leaves from neighbouring species. The within cluster STRESS, caused by the zero distance for a cluster point with itself, is relatively high. Looking at Figure 6.1 it seems that CDS is quite successful in keeping the leaves from the same species in the same cluster. We further see that species *a* is quite different from the others, and that *b* and *c* have leaves that seem to be more overlapping.

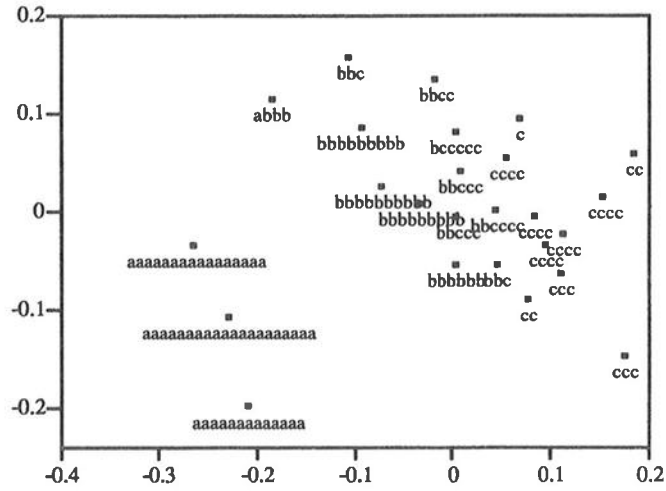


Figure 6.1 *Plot of coordinates of 25 clusters from CDS on distance matrix derived from the four descriptive variables of the Iris data. The clusters are labelled by the species of the leaves.*

6.1.4 Journal to journal citation data

A second interesting example is the analysis of citation data by scientometric maps, e.g., see Tijssen (1992). The basic idea is that articles in a scientific journal in a specific field tend to cite articles in other journals in the same field. In this way a map can be obtained to find groups of journals that are cognitively linked. Clearly, adding clusters has the advantage of finding distinct groups. We use the gravity model

$$d_{ij}^2(\mathbf{X}) = \frac{m_{i+}m_{+j}}{n_{ij}}, \tag{6.20}$$

with n_{ij} the number of citations between journal i and j, m_{i+} the sum of the citations to journal i. The gravity model is known from the relation between large masses in physics, like moon and earth, their distance and the gravity force. Thus, the gravity force n_{ij} between two journals is inversely related to their distance, given their masses m_{i+} and m_{+j}. Note that the gravity model corrects for the total number of citations of a journal. Clearly, if a journal is cited very often the probability of another journal citing the first one is larger. This model has been used in a similar context by Zielman (1991). Model (6.20) translates easily into the framework of STRESS as

$$\sigma^2(\mathbf{X}) = \sum_{ij} w_{ij}\left(\sqrt{\frac{m_{i+}m_{+j}}{n_{ij}}} - d_{ij}(\mathbf{X})\right)^2. \qquad (6.21)$$

Setting $\delta_{ij} = \sqrt{m_{i+}m_{+j}/n_{ij}}$ translates the gravity model back to STRESS. If n_{ij} is zero, δ_{ij} is not defined and w_{ij} is set to zero, otherwise w_{ij} is set to one. We may perform CDS to obtain clusters of journals.

The citation data stem from 72 internationally renowned journals in the field of earth sciences. The original table gives the number of citations in journal i to journal j which may differ from the number of citation in journal j to journal i. Here, we are interested in the symmetric part of the data only, so n_{ij} represents the average number of citations of journal i in j and vice versa. Furthermore, the number of self-citations were excluded, since they do not contribute to the structure of citations among journals.

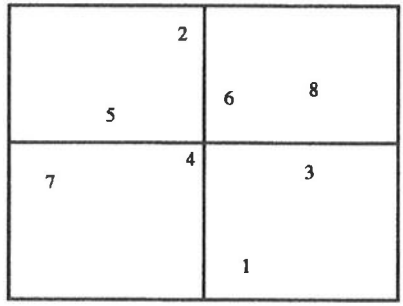


Figure 6.2 *Two dimensional representation of the 8 clusters of citation data of journals in earth sciences.*

Table 6.3 *Decomposition of STRESS for CDS with 8 clusters on citation data of journals in earth sciences.*

Source of STRESS	STRESS	% total STRESS
Between cluster object	0.099943	80.2
Within cluster object	0.017214	13.8
Between cluster	0.007478	6.0
Within cluster	0.0	0.0
Total	0.124635	100.0

Table 6.4 *Clusters obtained by CDS on citation data of journals in earth sciences. Journals titles are abbreviated.*

Cluster 1	Cluster 2	Cluster 3	Cluster 4
Am Mineral	J Biogeogr	Am J Sci	Aust J Eart
Chem Geol	J Paleontol	B Soc Geol	Coast Eng
Contr Min p	Lethaia	Cr Ac S Ii	Compt Rend
Earth Plan	Micropaleon	Earth Sci r	Geo-mar Let
Econ Geol b	Palaeogeo p	Geol Rundsc	Geol Foren
Geoch Cos a	Palaeontol	J Geology	Ian Sss Geo
J Petrology	Paleobiol	J Struc Geo	Init R Deep
Min Deposit	Quat Sci R	T Geol S Sa	J Foramin r
Precamb Res	Quatern Res	Tectonophys	J Geol S In
	Rev Palae p		Okeanologiy
			P I A S-ear
			Phi T Roy a
			Radiocarbon
Cluster 5	**Cluster 6**	**Cluster 7**	**Cluster 8**
B Marin Sci	Boreas	Cont Shelf	Aapg Bull
Global Plan	Eclog Geol	Deep-sea A	Can J Earth
J Marine Bi	Geol Mag	J Geo Res-o	Geol S Am b
Mar Micropa	Geol Soc M	J Marine Re	Geology
Nature	J Hum Evol	Mar Chem	J Geol Soc
Neth J Sea	Marine Geol	Marine Biol	J Sed Petro
Oceanol Act	Phys E Plan		Sediment Ge
Org Geochem			Sedimentol
Science			Tectonics

The configuration of clusters of CDS on the journal to journal citation data is given in Figure 6.2, the STRESS decomposition in Table 6.3, and the classification of journals into clusters in Table 6.4. Some clusters have a clear interpretation, like cluster 7, that contains journals involved with oceanography, and cluster 2, that consists of journals in palaeontology. Cluster 5 also contains journals from oceanography, marine biology, and multi-disciplinary journals. Clusters 1, 3 , and 8 consist of journals from geology, geosciences and geography, that may be considered to be the "harder" geology sciences. The interpretation of other clusters is sometimes difficult, like clusters 4 and 6, which seem to have journals that cite in an interdisciplinary fashion. The advantage of CDS is that from the positioning of the clusters in the configuration, something can be said about the (dis)similarity of the clusters, while in conventional non-hierarchical cluster analysis this information is frequently lost. It seems, for example, that cluster 1 has some similarity with clusters 3 and 8 (as confirmed above). Furthermore, from Figure 6.2 we see that the clusters involved in oceanography and palaeontology (clusters 2, 5 ,and 7) are located on the upper left side, and that interdisciplinary journals (clusters 4 and 6) are in the middle. These are the most striking aspects of journals in earth sciences considered here, that could be revealed by CDS.

6.2 CDS with fuzzy clusters

The cluster restrictions used above defines crisp clusters; an object belongs to one cluster only. Sometimes this restriction is too rigid. Moreover, CDS with crisp clusters is more prone to local minima as a result of the strong cluster constraints. In a less constrained model the loss of an object is due to all clusters, with a weight between zero and one and the weights per object summing to one. In the cluster analysis literature the use of this type of constraints is termed fuzzy clustering. In this section we develop a version of cluster differences scaling with fuzzy cluster membership. It is shown that under certain conditions, fuzzy CDS reduces to the crisp CDS discussed in the previous sections.

Similar to crisp CDS, we can describe the problem as one of minimizing the loss function

$$\sigma^2(\mathbf{G},\mathbf{X}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K}\sum_{l=1}^{K} g_{ik}^q g_{jl}^q w_{ij}\left(\delta_{ij} - d_{kl}(\mathbf{X})\right)^2, \tag{6.22}$$

where $\mathbf{X}$ is a $K\times p$ matrix of coordinates of K clusters, g_{ik}^q is a value denoting the strength of the cluster membership of object i to cluster k, and q is a fixed fuzzy clustering parameter. As indicated above, the main characteristic of fuzzy clustering is that the elements g_{ik} of the $n\times K$ matrix $\mathbf{G}$ can have any value between zero and one, provided they sum to one for each object i, i.e., $0 \leq g_{ik} \leq 1$ and $\sum_k g_{ik} = 1$. Thus, the STRESS for object pair ij is caused by a weighted sum of the membership of object i to cluster k times the membership of object j to cluster l. When comparing crisp CDS with fuzzy CDS we have lost the simple interpretation of an object belonging to one cluster only and obtained a large number of extra parameters to be estimated. However, the fuzzy cluster membership parameters allow for a more diffuse object cluster relationship.

A convergent algorithm for fuzzy CDS can be obtained by minimizing $\sigma^2(\mathbf{G},\mathbf{X})$ alternatingly over $\mathbf{X}$ and $\mathbf{G}$. We first consider the minimization of (6.22) over $\mathbf{X}$ for fixed $\mathbf{G}$. We start by rewriting (6.22) as

$$\begin{aligned}
\sigma^2(\mathbf{G},\mathbf{X}) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K}\sum_{l=1}^{K} g_{ik}^q g_{jl}^q w_{ij}\left(\delta_{ij}^2 + d_{kl}^2(\mathbf{X}) - 2\delta_{ij}\, d_{kl}(\mathbf{X})\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{K}\sum_{l=1}^{K} g_{ik}^q g_{jl}^q w_{ij}\delta_{ij}^2 + \sum_{k=1}^{K}\sum_{l=1}^{K} d_{kl}^2(\mathbf{X}) \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ik}^q g_{jl}^q w_{ij} \\
&\quad - 2\sum_{k=1}^{K}\sum_{l=1}^{K} d_{kl}(\mathbf{X}) \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ik}^q g_{jl}^q w_{ij}\delta_{ij} \\
&= \eta_\delta^2(\mathbf{G}) + \eta^2(\mathbf{G},\mathbf{X}) - 2\rho(\mathbf{G},\mathbf{X}).
\end{aligned} \tag{6.23}$$

Let $\mathbf{A}(\mathbf{G})$ be the matrix with off diagonal elements $a_{kl} = -\sum_{i,j}^{n} g_{ik}^q g_{jl}^q w_{ij}$ if $k \neq l$ and diagonal elements $a_{kk} = -\sum_l a_{kl}$. Then we have $\eta^2(\mathbf{G},\mathbf{X}) = \text{tr } \mathbf{X}'\mathbf{A}(\mathbf{G})\mathbf{X}$. Due to majorization we know that

$$-\rho(\mathbf{G},\mathbf{X}) = -\hat{\rho}(\mathbf{G},\mathbf{X},\mathbf{X}) \leq -\hat{\rho}(\mathbf{G},\mathbf{X},\mathbf{Y}) = -\text{tr } \mathbf{X}'\mathbf{B}(\mathbf{G},\mathbf{Y})\mathbf{Y} \tag{6.24}$$

where $\mathbf{B}(\mathbf{G},\mathbf{Y})$ is a matrix with off diagonal elements $b_{kl} = -\sum_{i<j}^{n} g_{ik}^q g_{jl}^q w_{ij} \delta_{ij}/d_{ij}(\mathbf{Y})$ if $d_{ij}(\mathbf{Y}) \neq 0$, $b_{kl} = 0$ if $d_{ij}(\mathbf{Y}) = 0$, and diagonal elements $b_{kk} = -\sum_l b_{kl}$. This allows us to majorize $\sigma^2(\mathbf{G},\mathbf{X})$ by

$$\begin{aligned} \hat{\sigma}^2(\mathbf{G},\mathbf{X},\mathbf{Y}) &= \eta_\delta^2(\mathbf{G}) + \eta^2(\mathbf{G},\mathbf{X}) - 2\hat{\rho}(\mathbf{G},\mathbf{X},\mathbf{Y}) \\ &= \eta_\delta^2(\mathbf{G}) + \text{tr } \mathbf{X}'\mathbf{A}(\mathbf{G})\mathbf{X} - 2\text{tr } \mathbf{X}'\mathbf{B}(\mathbf{G},\mathbf{Y})\mathbf{Y}. \end{aligned} \tag{6.25}$$

Setting the gradient of $\hat{\sigma}^2(\mathbf{G},\mathbf{X},\mathbf{Y})$ equal to $\mathbf{0}$ gives the update

$$\mathbf{X}^+ \leftarrow \mathbf{A}(\mathbf{G})^-\mathbf{B}(\mathbf{G},\mathbf{Y})\mathbf{Y}. \tag{6.26}$$

By majorization, (6.26) produces a convergent series of nonincreasing STRESS values.

The second step of the algorithm is to minimize $\sigma^2(\mathbf{G},\mathbf{X})$ over $\mathbf{G}$, for fixed $\mathbf{X}$. In fact, we may minimize one row i of $\mathbf{G}$ at a time. Let $\sigma_i^2(\mathbf{G},\mathbf{X})$ denote the contribution of object i to STRESS

$$\sigma_i^2(\mathbf{G},\mathbf{X}) = \sum_{k=1}^{K} g_{ik}^q \sum_{j=1}^{n} \sum_{l=1}^{K} g_{jl}^q w_{ij} \left(\delta_{ij} - d_{kl}(\mathbf{X})\right)^2 = \sum_{k=1}^{K} g_{ik}^q \gamma_{ik} \tag{6.27}$$

where $\gamma_{ik} = \sum_{j=1}^{n} \sum_{l=1}^{K} g_{jl}^q w_{ij} \left(\delta_{ij} - d_{kl}(\mathbf{X})\right)^2$ and g_{ik} restricted to the constraints indicated above. Note that it is known in the literature how this can be done, but we give an independent proof here, i.e., we prove that choosing

$$g_{ik} = \frac{(\gamma_{ik})^{-1/(q-1)}}{\sum_{l=1}^{K} (\gamma_{il})^{-1/(q-1)}} \tag{6.28}$$

gives the optimal cluster membership for fixed $\mathbf{X}$ and $q \geq 1$. A necessary condition for a minimum of $\sigma_i^2(\mathbf{G},\mathbf{X})$ under the constraint $\sum_s g_{is} = 1$ is that the first derivative of the Lagrangian function

$$\sigma_i^2(\mathbf{G},\mathbf{X}) - \lambda\left(\sum_{k=1}^{K} g_{ik} - 1\right) = \left(\sum_{k=1}^{K} \gamma_{ik} g_{ik}^q\right) - \lambda\left(\sum_{k=1}^{K} g_{ik} - 1\right) \tag{6.29}$$

to g_{is} and λ must be zero. Solving this system of equations gives

$$g_{ik} = \frac{\lambda^{1/(q-1)}}{\gamma_{ik}^{1/(q-1)}} \quad \text{and} \quad \sum_{k=1}^{K} g_{ik} = 1 \,, \tag{6.30}$$

which for $\lambda^{1/(q-1)} = (\sum_k \gamma_{ik}^{-1/(q-1)})^{-1}$ results in g_{ik} as defined in (6.28). It is not difficult to verify that $g_{ik} \geq 0$ and $\sum_k g_{ik} = 1$ and consequently the constraints hold. A similar result derived for fuzzy clustering is already known since Bezdek (1981). For large q the update for g_{ik} (6.28) gives approximately equal values almost irrespective of γ_{ik}, except for values of γ_{ik} that tend to zero. Therefore, it seems wise to choose q between 1 and 2.

As q approaches 1, the current procedure amounts to setting $g_{il} = 1$ if $\gamma_{il} = \min_k \gamma_{ik}$ and $g_{ik} = 0$ otherwise, which is exactly what is done for crisp clustering, discussed in the previous sections.

A serious drawback of fuzzy CDS is that a lot of extra cluster membership parameters are obtained if the number of clusters is large. However, choosing q close to one, e.g., 1.2, may reduce the number of interpretable values of g_{ik} per object. Furthermore, if the number of clusters is two or three, a simultaneous graphical representation of **G** and **X** can be obtained. For example, in case of three clusters, we can plot the cluster memberships inside the triangle formed by the cluster points, since **G** span a two dimensional triangle because **G1** = **1**.

Fuzzy CDS (6.22) can not be expressed as an ordinary MDS problem with the restriction that the configuration is of the form **GX** with **G1** = **1** and $0 \leq g_{ik} \leq 1$. However, in such an analysis we expect that the cluster configuration forms a hypersphere (or a circle in two dimensions), since all objects are weighted centroids of the cluster coordinates. These restrictions can be incorporated in the general restricted MDS framework by using Dykstra's cyclic projection algorithm (Dykstra ,1983). We do not pursue this approach here.

A particularly useful application of fuzzy CDS is presented in the next section.

6.3 Obtaining a good start configuration for CDS by using fuzzy CDS

Unreported numerical experiments by the author have revealed a severe local minimum problem with crisp CDS. Often we obtain a different local minimum by starting from a different initial clustering assignment of the objects. The reason for this can be understood by regarding CDS as an ordinary MDS with cluster restrictions on the configuration (see section 6.1.1). Thus, every configuration can be written as **GX** where **G** is the matrix that defines the cluster membership and **X** the matrix of cluster coordinates. All objects in the same cluster are restricted to have equal coordinates. These equality constraints give rise to the extra number of local minima in CDS.

In the previous section, we have seen that fuzzy CDS reduces to crisp CDS as q tends to 1. We can exploit this feature to help reduce the local minimum problem of crisp CDS by using the following algorithm to obtain a good start configuration for CDS. We start with a fuzzy CDS analysis with a high value of q, say $q = 2$. We can expect that all values

g_{ik} are approximately equal (cf. section 6.2). The resulting **G** and **X** are used as start configurations for a fuzzy CDS analysis with a slightly smaller q. The latter step is repeated for decreasing values of q approaching one. As q gets close to one, we start the crisp CDS algorithm and obtain our final solution. The main rationale behind the suggested procedure is that the equality constraints are gradually introduced, not at once. At the beginning there is a lot of freedom in the optimization process of choosing **G**, which diminishes as q gets smaller, and reduces to the strict equality constraints for $q = 1$.

To test this hypothesis, we did an experiment to compare this strategy (*a*) with two other strategies. The first competing strategy is to do CDS using a start configuration obtained from an ordinary MDS without any cluster constraints, followed by K-means clustering (strategy *b*). The second alternative strategy is to do CDS 10 times with a random start configuration **X** and random initial cluster assignment **G** and keep the best result (strategy *c*). We compared these three strategies for 5 and 10 clusters, 20 and 40 number of objects. For each combination, we generated 25 random dissimilarity matrices and noted the STRESS value for each strategy. For all 100 dissimilarity matrices generated, strategy *a* yielded the lowest STRESS value 92 times, strategy *b* 5 times, and *c* 3 times. A more detailed description of these results is given in Table 6.5. In the table we clearly see that strategy *a* is superior to the other strategies in many respects. It yields the best STRESS values in 92% of the cases, in all 50 cases with 10 clusters, and it has STRESS values close to the smallest ones in case it misses. The average difference with the lowest STRESS value shows the relative performance of the methods. Strategies *b* and *c* do not seem to differ much from each other.

Table 6.5 *Results of an experiment in which three strategies for generating a start configuration for CDS are compared. In each cell we report the number of times a strategy yielded the best STRESS values for CDS (top value), the mean STRESS value, and the average difference with the lowest STRESS value (in parenthesis).*

		Strategy *a* (fuzzy CDS)	Strategy *b* (MDS/K-means)	Strategy *c* (10 random starts)
5 clusters	$n = 20$	21	2	2
		17.80 (0.03)	18.99 (1.22)	18.73 (0.96)
	$n = 40$	21	3	1
		91.73 (0.10)	94.67 (3.07)	94.95 (3.33)
10 clusters	$n = 20$	25	0	0
		13.46 (0)	15.06 (1.59)	14.37 (0.91)
	$n = 40$	25	0	0
		74.49 (0)	78.15 (3.66)	78.18 (3.69)
Total		92	5	3

We may conclude that strategy *a* (using fuzzy CDS to generate a start configuration for **X** and **G**) is far superior over the other two strategies. It seems to be a good strategy to avoid local minima of CDS, since it out performs the multistart strategy *c*. However, we must note that strategy *a* is computationally intensive, especially for large *n* and a large number of clusters. This was the main reason for not including a level with a higher number of objects and more clusters in our experiment. For moderate size *n*, the strategy described here is very useful.

6.4 Concluding remarks

We have elaborated on a clustering method for MDS originally proposed by Heiser (1992). The aim was to facilitate interpretation by imposing cluster restrictions on the objects and still get a graphical representation of the clusters to model the dissimilarities. We indicated how the STRESS in CDS can be decomposed into four additive components of within and between cluster STRESS. We showed that CDS can be seen as ordinary MDS with cluster restrictions on the configuration. Furthermore, we generalized CDS to deal with fuzzy clusters. It was shown by an experiment that a start configuration for CDS based on repeated fuzzy clustering with decreasing fuzzy cluster parameter q, gave the lowest STRESS value. This is a particularly useful application of fuzzy CDS to obtain better local minima in crisp CDS.

CHAPTER 7

STRESS WITH MINKOWSKI DISTANCES

In the previous chapters, we limited the majorization method for multidimensional scaling to Euclidean distances only. Here, we extend the majorization algorithm to deal with Minkowski distances with $1 \leq q \leq 2$ and suggest an algorithm, which is partially based on majorization, for q outside this range. We give some convergence proofs and extend the zero distance theorem of De Leeuw (1984) to Minkowski distances with $q > 1$. We illustrate the algorithm by an example. The results presented here are for a large part published earlier in Groenen, Mathar, and Heiser (1992).

Most frequently, Euclidean distance is used in MDS, but this need not be so. An important family of distance measures is formed by the Minkowski distances of which the Euclidean distance is a special case. In the latter case, we simply can use the SMACOF algorithm from section 1.3. An attractive feature of SMACOF is that it produces a monotone nonincreasing sequence of values of the loss function by using the concept of iterative majorization. De Leeuw (1977) treats multidimensional scaling in the framework of convex analysis. He, for the first time, gave a convergence proof for Euclidean distances, and discussed extensions for general q, without giving an explicit algorithm. Using a similar approach, Mathar and Groenen (1991) give a convergence proof of a nested algorithm for MDS, assuming that the inner optimization problem has a unique solution. They also give interpretations in terms of directional derivatives and subgradient-projection methods. For $p = 2$ this algorithm was made explicit. Considering STRESS as a DC-function (difference of convex functions) Mathar and Meyer (1992) obtain subgradients for arbitrary q. This approach leads to an eigenvector problem that is solved by inverse iteration (see e.g., Peters and Wilkinson, 1971). Interestingly enough, in the Euclidean case this reduces to SMACOF again.

Here we extend the majorization approach to least squares scaling with Minkowski distances for $1 \leq q \leq 2$. For q outside this range, algorithms that are partially based on majorization are developed. Kruskal's (1964a,b) method also covers Minkowski distances but works with a complicated step-size procedure which makes convergence uncertain. We think that the majorization approach has at least two advantages. By definition it generates a nonincreasing sequence of STRESS values, which immediately yields convergence. Moreover, without applying gradients or subgradients, it gives insight into the behaviour of the STRESS surface by a local model at supporting points.

As elsewhere in this monograph, we minimize the STRESS function,

$$\sigma^2(\mathbf{X}) = \sum_{i<j}^{n} w_{ij}\left(\delta_{ij} - d_{ij}(\mathbf{X})\right)^2 , \tag{7.1}$$

over all configurations $\mathbf{X}$ (an $n \times p$ matrix with coordinates of n objects in p dimensions). However, throughout this chapter the distance

$$d_{ij}(\mathbf{X}) = \left(\sum_{s=1}^{p} \left| x_{is} - x_{js} \right|^q \right)^{1/q} , \quad 1 \le q \le \infty , \tag{7.2}$$

denotes the Minkowski distance. As usual, we assume that the quantities w_{ij} are fixed nonnegative weights, δ_{ij} are nonnegative dissimilarities and that the weight matrix is irreducible (see section 1.3). Some well known metrics and corresponding norms are obtained by proper choices of q, like the city-block metric (or Manhattan metric) for $q = 1$, the Euclidean for $q = 2$ and the max norm for $q = \infty$. For a recent review article on Minkowski distances in multidimensional scaling we refer to Arabie (1991).

Let us rewrite (7.1) in the familiar form

$$\begin{aligned} \sigma^2(\mathbf{X}) &= \sum_{i<j}^{n} w_{ij}\delta_{ij}^2 + \sum_{i<j}^{n} w_{ij}\, d_{ij}^2(\mathbf{X}) - 2\sum_{i<j}^{n} w_{ij}\delta_{ij}\, d_{ij}(\mathbf{X}) \\ &= \eta_\delta^2 + \eta^2(\mathbf{X}) - 2\rho(\mathbf{X}). \end{aligned} \tag{7.3}$$

We shall see that for $1 \le q \le 2$ a convergent algorithm for minimizing (7.3) can be obtained by using majorization. For $q < 1$ or $q > 2$ a convergent majorization algorithm is obtained by using an inner minimization step of a convex function. However, in the case of unidimensional scaling, $p = 1$, the Minkowski distance is independent of q and gradient based algorithms run into a local minimum within a few steps. Here, more powerful combinatorial optimization methods are available (see section 2.2).

In the next section we apply the principle of majorization to $-\rho(\mathbf{X})$ and $\eta^2(\mathbf{X})$. From this result a convergent majorization algorithm for $1 \le q \le 2$ is derived and some convergence properties are given. Then, we discuss two algorithms for q values outside this range. Next, we show differentiability of STRESS at a local minimum for $q > 1$. Finally, as an illustration of our procedure, we present a small example and compare our algorithm to the one of Kruskal (1964a,b).

7.1 Majorizing STRESS

Here we propose majorizing functions for the separate parts of $\sigma^2(\mathbf{X})$ in (7.3). We show that for certain ranges of q, $-\rho(\mathbf{X})$ can be majorized by linear majorization and $\eta^2(\mathbf{X})$ by quadratic majorization. For a more detailed discussion of minimizing a function by

iterative majorization, we refer to section 1.1. Note that a majorization algorithm stops at any point where the necessary condition for a stationary point is satisfied. Other information has to be used to check if the point is a global minimum, a local minimum or even a saddle point. If we want to find the global minimum, we could use any of the global optimization methods discussed in chapters 2 and 3.

7.1.1 Majorization of $-\rho(\mathbf{X})$

The cross product term $-\rho(\mathbf{X})$ in (7.3) is majorized by applying Hölder's inequality to $-d_{ij}(\mathbf{X})$ for $q \geq 1$. If the denominator is positive, then for any $\mathbf{X}$ and $\mathbf{Y}$,

$$\begin{aligned} -d_{ij}(\mathbf{X}) = -\left(\sum_{s=1}^{p}\left|x_{is}-x_{js}\right|^{q}\right)^{1/q} &\leq -\frac{\sum_{s=1}^{p}\left|x_{is}-x_{js}\right|\left|y_{is}-y_{js}\right|^{q-1}}{\left(\sum_{s=1}^{p}\left|y_{is}-y_{js}\right|^{q}\right)^{(q-1)/q}} \\ &\leq -\frac{\sum_{s=1}^{p}\left(x_{is}-x_{js}\right)\left(y_{is}-y_{js}\right)\left|y_{is}-y_{js}\right|^{q-2}}{\left(\sum_{s=1}^{p}\left|y_{is}-y_{js}\right|^{q}\right)^{(q-1)/q}}, \end{aligned} \tag{7.4}$$

with equality if $x_{is} = y_{is}$ for all i and s. The second part of the inequality (7.4) holds, because $-|x_{is}-x_{js}||y_{is}-y_{js}|^{q-1} \leq -(x_{is}-x_{js})(y_{is}-y_{js})|y_{is}-y_{js}|^{q-2}$. If $d_{ij}(\mathbf{Y}) = 0$ we simply define the right hand side as 0, which preserves the validity of inequality (7.4). Furthermore, we use the conventions $0^0 = 1$ and $0.* = 0$. Multiplying both sides of (7.4) by $w_{ij}\delta_{ij}$ and summing over all $i<j$ we obtain the following inequality

$$-\rho(\mathbf{X}) \leq -\sum_{s=1}^{p}\mathbf{x}_s'\mathbf{B}_s(\mathbf{Y})\,\mathbf{y}_s = -\hat{\rho}(\mathbf{X},\mathbf{Y}), \tag{7.5}$$

where $\mathbf{x}_s$ and $\mathbf{y}_s$ denote the columns of $\mathbf{X}$ and $\mathbf{Y}$, respectively. $\mathbf{B}_s(\mathbf{Y})$ has off-diagonal elements

$$b_{ij}^{(s)} = -\frac{w_{ij}\delta_{ij}\left|y_{is}-y_{js}\right|^{(q-2)}}{d_{ij}^{q-1}(\mathbf{Y})}, \quad i \neq j, \tag{7.6}$$

if $d_{ij}(\mathbf{Y}) > 0$ and $b_{ij}^{(s)} = 0$ otherwise, and diagonal elements $b_{ii}^{(s)} = -\sum_{j\neq i} b_{ij}^{(s)}$. Obviously $\rho(\mathbf{X}) = \hat{\rho}(\mathbf{X},\mathbf{X})$ holds for all $\mathbf{X}$.

7.1.2 *Majorization of* $\eta^2(\mathbf{X})$

Now, consider $\eta^2(\mathbf{X})$ and more specifically its elements $d_{ij}^2(\mathbf{X})$. This expression is the square of the Minkowski distance, which can be rewritten as

$$d_{ij}^2(\mathbf{X}) = \left(\sum_{s=1}^{p} |x_{is} - x_{js}|^q\right)^{2/q} = \left(\sum_{s=1}^{p} |x_{is} - x_{js}|^{2r}\right)^{1/r} \tag{7.7}$$

for $r = q/2$. Using Hölder's inequality for $r \leq 1$, thus $q \leq 2$, gives

$$d_{ij}^2(\mathbf{X}) = \left(\sum_{s=1}^{p} |x_{is} - x_{js}|^{2r}\right)^{1/r} \leq \frac{\sum_{s=1}^{p} (x_{is} - x_{js})^2 \, |y_{is} - y_{js}|^{2(r-1)}}{\left(\sum_{s=1}^{p} |y_{is} - y_{js}|^{2r}\right)^{(r-1)/r}}, \tag{7.8}$$

for all positive $|y_{is} - y_{js}|$ with equality if $x_{is} = y_{is}$ for all i and s. Since $\mathbf{x}'\mathbf{A}\mathbf{y} = \sum_{i<j} a_{ij}(x_{is} - x_{js})(y_{is} - y_{js})$ holds for any symmetric matrix $\mathbf{A}$ having off-diagonal elements $-a_{ij}$ and diagonal elements $\sum_{j \neq i} a_{ij}$, after multiplication of both sides of (7.8) by w_{ij} we get the following expression

$$\eta^2(\mathbf{X}) \leq \sum_{s=1}^{p} \mathbf{x}_s' \mathbf{A}_s(\mathbf{Y}) \, \mathbf{x}_s = \hat{\eta}^2(\mathbf{X}, \mathbf{Y}), \tag{7.9}$$

where $\mathbf{A}_s(\mathbf{Y})$ has off-diagonal elements

$$a_{ij}^{(s)} = -\frac{w_{ij} |y_{is} - y_{js}|^{(q-2)}}{d_{ij}^{q-2}(\mathbf{Y})}, \quad i \neq j, \tag{7.10}$$

and diagonal elements $a_{ii}^{(s)} = -\sum_{j \neq i} a_{ij}^{(s)}$. Using the same conventions as above we obtain $\eta^2(\mathbf{X}) = \hat{\eta}^2(\mathbf{X}, \mathbf{X})$ for any configuration $\mathbf{X}$.

If $y_{is} - y_{js} = 0$ for some i,j,s, inequality (7.8) may not be true. In that case we simply replace $(y_{is} - y_{js})^2$ by some small positive constant ε, in much the same way as Heiser (1991) treats this when dealing with negative dissimilarities. Note that we needed the same adaptation in the tunneling algorithm of chapter 3. The majorizing function remains larger than $\eta^2(\mathbf{X})$, but does not touch $\eta^2(\mathbf{X})$ for $\mathbf{X} = \mathbf{Y}$, although we may get arbitrarily close by letting ε approach zero.

The majorization inequality (7.9) derived here holds for any $q \leq 2$ and the majorizing function $\hat{\eta}^2(\mathbf{X}, \mathbf{Y})$ is a quadratic function in $\mathbf{X}$. For $q > 2$ all inequalities in this section have reversed sign.

7.2 The majorization algorithm for $1 \le q \le 2$

The two majorization inequalities can be used simultaneously for $1 \le q \le 2$. This strategy results in an algorithm that produces a monotone decreasing series of function values and hence is convergent in this sense. Note that at best a local minimum is reached.

The majorization algorithm is derived from the stationary equation of the majorizing function, which is quadratic in $\mathbf{X}$. Thus, we have

$$\begin{aligned}\sigma^2(\mathbf{X}) &\le \eta_\delta^2 + \hat{\eta}^2(\mathbf{X},\mathbf{Y}) - 2\hat{\rho}(\mathbf{X},\mathbf{Y}) \\ &= \eta_\delta^2 + \textstyle\sum_{s=1}^{p} \mathbf{x}_s'\mathbf{A}_s(\mathbf{Y})\mathbf{x}_s - 2\sum_{s=1}^{p} \mathbf{x}_s'\mathbf{B}_s(\mathbf{Y})\mathbf{y}_s. \end{aligned} \tag{7.11}$$

Setting the gradient of the majorizing function (7.11) equal to zero implies for all s

$$\mathbf{x}_s = \mathbf{A}_s(\mathbf{Y})^{-}\mathbf{B}_s(\mathbf{Y})\mathbf{y}_s\,, \tag{7.12}$$

where $\mathbf{A}_s(\mathbf{Y})^{-}$ is any generalized inverse of $\mathbf{A}_s(\mathbf{Y})$. The update may be computed simultaneously or dimensionwise. It can be shown that for $q = 2$, the proposed algorithm with update (7.12) simply reduces to the SMACOF algorithm.

Fortunately, almost all convergence theorems known from SMACOF still hold. Here we follow the proofs of De Leeuw (1988). The notation is simplified whenever it can be done without introducing ambiguity; at iteration m we write $\sigma_m^2 = \sigma^2(\mathbf{X}^m)$, $\rho_m = \rho(\mathbf{X}^m)$, $\eta_m^2 = \eta^2(\mathbf{X}^m)$, $\mathbf{B}_s^m = \mathbf{B}_s(\mathbf{X}^m)$, $\mathbf{A}_s^m = \mathbf{A}_s(\mathbf{X}^m)$. For convenience, we assume without loss of generality, $\eta_\delta^2 = 1$. Observe that by applying the above ε-procedure, all $\mathbf{A}_s^m$ may be assumed to have rank $n - 1$ (cf. Mathar and Meyer, 1992).

Since STRESS is the sum of squared differences, we may use the Cauchy-Schwarz inequality to obtain $\rho_m \le \eta_m$. Because of $\mathbf{B}_s^m = \mathbf{A}_s^m\mathbf{A}_s^{m-}\mathbf{B}_s^m$, by Cauchy-Schwarz we get $\sum_s \mathbf{x}_s^{m\prime}\mathbf{B}_s^m\mathbf{x}_s^m \le (\sum_s \mathbf{x}_s^{m\prime}\mathbf{A}_s^m\mathbf{x}_s^m)^{1/2}\,(\sum_s \mathbf{x}_s^{m+1\prime}\mathbf{A}_s^m\mathbf{x}_s^{m+1})^{1/2}$, or

$$\rho(\mathbf{X}^m) \le \eta(\mathbf{X}^m)\,\hat{\eta}(\mathbf{X}^{m+1},\mathbf{X}^m)\,. \tag{7.13}$$

Furthermore, we have

$$\eta^2(\mathbf{X}^{m+1}) \le \hat{\eta}^2(\mathbf{X}^{m+1},\mathbf{X}^m) = \textstyle\sum_s \mathbf{x}_s^{m\prime}\mathbf{B}_s^m\mathbf{A}_s^{m-}\mathbf{B}_s^m\,\mathbf{x}_s^m = \hat{\rho}(\mathbf{X}^{m+1},\mathbf{X}^m) \le \rho(\mathbf{X}^{m+1})\,, \tag{7.14}$$

which holds because of the majorization inequalities. Combining (7.13) and (7.14) gives

$$\eta(\mathbf{X}^m) \le \frac{\rho(\mathbf{X}^m)}{\eta(\mathbf{X}^m)} \le \hat{\eta}(\mathbf{X}^{m+1},\mathbf{X}^m)\,. \tag{7.15}$$

These inequalities let us form the following chain:

$$\eta_m^2 \le \rho_m \le \eta_m\,\hat{\eta}(\mathbf{X}^{m+1},\mathbf{X}^m) \le \hat{\eta}^2(\mathbf{X}^{m+1},\mathbf{X}^m) = \hat{\rho}(\mathbf{X}^{m+1},\mathbf{X}^m) \le \rho_{m+1} \le \eta_{m+1} \le 1. \tag{7.16}$$

Further, define the measure of difference in squared distances as

$$\begin{aligned}\varepsilon_m^2 &= \hat{\eta}^2(\mathbf{X}^m - \mathbf{X}^{m+1},\mathbf{X}^m) = \sum_s (\mathbf{x}_s^m - \mathbf{A}_s^{m-}\mathbf{B}_s^m\mathbf{x}_s^m)'\mathbf{A}_s^m\,(\mathbf{x}_s^m - \mathbf{A}_s^{m-}\mathbf{B}_s^m\,\mathbf{x}_s^m)\\ &= \eta^2(\mathbf{X}^m) + \hat{\eta}^2(\mathbf{X}^{m+1},\mathbf{X}^m) - 2\rho(\mathbf{X}^m). \qquad (7.17)\end{aligned}$$

These inequalities lead to the following observations:

1. $\rho_m \uparrow \rho_\infty$,
2. $\eta_m^2 \uparrow \eta_\infty^2 = \rho_\infty$,
3. $\sigma_m^2 \downarrow \sigma_\infty^2 = 1 - \rho_\infty$,
4. $\varepsilon_m^2 \to 0$.

The last assertion is proved by filling in the limiting values of η_m^2, $\hat{\eta}^2(\mathbf{X}^{m+1},\mathbf{X}^m)$ and ρ_m. Since the metric of ε_m^2 depends on m, the convergence of ε_m^2 is hard to interpret with the exception of $q = 2$ when $\mathbf{A}_s^m$ is a matrix that is fixed and does not depend on $\mathbf{X}^m$. Although these observations are comforting, we do not have proofs about the convergence behaviour of $\mathbf{X}^m$, except for the case $q = 2$, which are given by De Leeuw (1988).

For $q = 2$, the current algorithm has a linear convergence rate at isolated local minima (see De Leeuw, 1988). Due to the convergence theorem 1.1 given in chapter 1, we also expect that the present majorization algorithm for $1 \leq q \leq 2$ has a linear convergence rate. This can be seen as follows. To establish the convergence rate of the quadratic majorization algorithm, we have to find the largest eigenvalue of the derivative of the mapping (7.12) at a local minimum $\mathbf{X}^*$ conforming to theorem 1.1. Using similar arguments as in theorem 1.1 it can be shown that the derivative of (7.12), if it exists, equals $\mathbf{A}(\mathbf{X}^*)^-\nabla^2\rho(\mathbf{X}^*)$ and has real eigenvalues. Moreover, the eigenvalues are nonnegative, because the eigensystem can be written equivalently as $\nabla^2\rho(\mathbf{X}^*)\mathbf{z} = \lambda\mathbf{A}(\mathbf{X}^*)\mathbf{z}$, where $\nabla^2\rho(\mathbf{X}^*)$ is positive semi-definite because $\rho(\mathbf{X}^*)$ is a convex function, and $\mathbf{z}$ is an eigenvector. Consider the Hessian $\nabla^2\sigma^2(\mathbf{X}^*)$ of $\sigma^2(\mathbf{X})$ at a local minimum $\mathbf{X}^*$. Then $\nabla^2\sigma^2(\mathbf{X}^*) = \nabla^2\eta^2(\mathbf{X}^*) - 2\nabla^2\rho(\mathbf{X}^*)$ must be positive semi-definite. Due to quadratic majorization of $\eta^2(\mathbf{X})$, we know that $\text{vec}(\mathbf{X})'\nabla^2\eta^2(\mathbf{X})\text{vec}(\mathbf{X}) \leq \sum_s \mathbf{x}_s'\mathbf{A}_s(\mathbf{Y})\mathbf{x}_s = \text{vec}(\mathbf{X})'\mathbf{A}(\mathbf{Y})\text{vec}(\mathbf{X})$, where $\mathbf{A}(\mathbf{Y})$ is a blockdiagonal matrix with diagonal blocks $\mathbf{A}_s(\mathbf{Y})$. This implies that $\nabla^2\eta^2(\mathbf{X}^*)$ has eigenvalues that are smaller than or equal to those of $2\mathbf{A}(\mathbf{X}^*)$. Consequently, $\mathbf{A}(\mathbf{X}^*) - \nabla^2\rho(\mathbf{X}^*)$ is also positive semi-definite, or, equivalently $\mathbf{I} - \mathbf{A}(\mathbf{X}^*)^-\nabla^2\rho(\mathbf{X}^*)$ is positive semi-definite. Therefore, the largest eigenvalue λ of $\mathbf{A}(\mathbf{X}^*)^-\nabla^2\rho(\mathbf{X}^*)$ must have $0 \leq \lambda \leq 1$. Theorem 1.1 tells that if $\mathbf{X}$ converges to $\mathbf{X}^*$, then we have linear convergence. If λ is smaller than 1, the convergence rate is equal to λ.

Near a local minimum many functions (including STRESS) tend to behave like a quadratic function. Since we only use first order information, i.e., gradient information if STRESS is differentiable, our algorithm can be viewed as a steepest descent algorithm. Such algorithms are known to have orthogonal subsequent search directions near the local

minimum that causes slow convergence, especially for almost quadratic functions. To avoid this undesirable behaviour, De Leeuw and Heiser (1980) proposed the use of a relaxed update $\mathbf{X}^+ = (1-\alpha)\mathbf{X} + \alpha\overline{\mathbf{X}}$ with $0 \leq \alpha \leq 2$, and $\overline{\mathbf{X}}$ is the update as defined in (7.12). They reported that a fixed value of $\alpha = 2$ approximately halved the number of iterations. Here, we prove that the relaxed update also retains convergence for the general algorithm proposed above.

Let us start by noting that STRESS may be written alternatively as

$$\sigma^2(\mathbf{X}) = \eta_\delta^2 + \hat{\eta}^2(\mathbf{X} - \overline{\mathbf{X}}, \mathbf{X}) - \hat{\eta}^2(\overline{\mathbf{X}}, \mathbf{X}). \tag{7.18}$$

Further, using $\sigma^2(\mathbf{X}^+) \leq \hat{\sigma}^2(\mathbf{X}^+, \mathbf{X})$ and some reformulation we have

$$\begin{aligned}
\sigma^2(\mathbf{X}^+) &\leq \eta_\delta^2 + \hat{\eta}^2((1-\alpha)\mathbf{X} + \alpha\overline{\mathbf{X}}, \mathbf{X}) - 2\hat{\rho}((1-\alpha)\mathbf{X} + \alpha\overline{\mathbf{X}}, \mathbf{X}) \\
&= \eta_\delta^2 + (1-\alpha)^2\hat{\eta}^2(\mathbf{X} - \overline{\mathbf{X}}, \mathbf{X}) - \hat{\eta}^2(\overline{\mathbf{X}}, \mathbf{X}) \\
&= \eta_\delta^2 + \hat{\eta}^2(\mathbf{X} - \overline{\mathbf{X}}, \mathbf{X}) - \hat{\eta}^2(\overline{\mathbf{X}}, \mathbf{X}) + \alpha(\alpha-2)\hat{\eta}^2(\mathbf{X} - \overline{\mathbf{X}}, \mathbf{X}) \\
&= \sigma^2(\mathbf{X}) + \alpha(\alpha-2)\hat{\eta}^2(\mathbf{X} - \overline{\mathbf{X}}, \mathbf{X}).
\end{aligned} \tag{7.19}$$

So for $0 \leq \alpha \leq 2$, the relaxed update indeed reduces STRESS.

7.3 Minkowski distances with $q < 1$ or $q > 2$

De Leeuw (1977) discusses how STRESS for general Minkowski distances could be handled, but he indicates that generally there is a nontrivial inner optimization problem. Mathar and Groenen (1991) gave an algorithm for the maximization of a convex function over a convex set to which the problem of minimizing STRESS can be restated. Here, we elaborate on the majorization approach. Clearly, for q outside the range [1,2] the majorizing function (7.11) is not valid anymore. Nevertheless, we could still use it as a local quadratic approximation of STRESS and use the update defined by (7.12). Although the convergence results are no longer valid, we might end up with a solution satisfying the stationary equations. However, since we wish to retain convergence, we use the majorization inequalities, which suggest different approaches for $q < 1$ and $q > 2$.

7.3.1 Minkowski distances with $q > 2$

For any $q \geq 2$ we can majorize STRESS by

$$\sigma^2(\mathbf{X}) \leq \eta_\delta^2 + \eta^2(\mathbf{X}) - 2\hat{\rho}(\mathbf{X}, \mathbf{Y}). \tag{7.20}$$

The squared Minkowski distance is a convex function, hence so is $\eta^2(\mathbf{X})$. Consequently, the majorizing function (7.20), which is the sum of a convex function and a linear function, is convex itself. The basic majorization algorithm can be applied, i.e., find iteratively the minimum over $\mathbf{X}$ of the majorizing function. Here, this amounts to finding the minimum of a convex function, which may be done by standard (convex) optimization techniques. The updates are given by

$$\mathbf{X}^{m+1} = \underset{\mathbf{X}}{\operatorname{argmin}} \, (\eta^2(\mathbf{X}) - 2\,\hat{\rho}(\mathbf{X},\mathbf{X}^m)), \tag{7.21}$$

and because of majorization inequality (7.20) a nonincreasing sequence of function values $\sigma^2(\mathbf{X}^m)$ is obtained.

To derive convergence results, we need an expression for the stationary equation for a minimum point of (7.21). Note that if (7.20) is differentiable at $\mathbf{X}^{m+1}$, the gradients of $\eta^2(\mathbf{X})$ and $\hat{\rho}(\mathbf{X},\mathbf{X}^m)$ at $\mathbf{X}^{m+1}$ are given by

$$2\mathbf{A}_s(\mathbf{X}^{m+1})\mathbf{x}_s^{m+1} \quad \text{and} \quad \mathbf{B}_s(\mathbf{X}^m)\mathbf{x}_s^m, \tag{7.22}$$

respectively. Thus, the stationary equation for a minimum point of (7.21) is $\mathbf{A}(\mathbf{X}^{m+1})\mathbf{x}_s^{m+1} = \mathbf{B}(\mathbf{X}^m)\mathbf{x}_s^m$ for all s, or equivalently

$$\mathbf{x}_s^{m+1} = \mathbf{A}_s(\mathbf{X}^{m+1})^{-}\mathbf{B}_s(\mathbf{X}^m)\mathbf{x}_s^m. \tag{7.23}$$

The convergence proof is based on (7.23) and follows the arguments in the previous section closely. Here too we have $\rho_m \leq \eta_m$. Since by Cauchy-Schwarz $\sum_s \mathbf{x}_s^{m\prime}\mathbf{B}_s^m\mathbf{x}_s^m \leq (\sum_s \mathbf{x}_s^{m\prime}\mathbf{A}_s^{m+1}\mathbf{x}_s^m)^{1/2}\,(\sum_s \mathbf{x}_s^{m+1\prime}\mathbf{A}_s^{m+1}\mathbf{x}_s^{m+1})^{1/2}$, or

$$\rho(\mathbf{X}^m) \leq \hat{\eta}(\mathbf{X}^m,\mathbf{X}^{m+1})\eta(\mathbf{X}^{m+1}) \leq \eta_m\eta_{m+1}. \tag{7.24}$$

The last inequality follows from (7.9) with reversed inequality sign for $q > 2$. Finally, we need

$$\eta^2(\mathbf{X}^{m+1}) = \hat{\rho}(\mathbf{X}^{m+1},\mathbf{X}^m) \leq \rho(\mathbf{X}^{m+1}), \tag{7.25}$$

which follows from majorization (7.5). (7.24) and (7.25) yield $\eta_m \leq \rho_m/\eta_m \leq \eta_{m+1}$ and $\eta_m \leq 1$. Altogether we have the chain

$$\eta_m^2 \leq \rho_m \leq \eta_{m+1}\eta_m \leq \eta_{m+1}^2 \leq \rho_{m+1} \leq \eta_{m+1} \leq 1, \tag{7.26}$$

that proves the convergence properties of ρ_m, η_m^2 and σ_m^2. We can also obtain a converging sequence of ε_m^2 by defining its metric to be $\mathbf{A}_s^{m+1}$ for each dimension s. A relaxed version of algorithm (7.21) can be obtained by searching for a configuration which has lower value of the majorizing function in (7.20), but not necessarily the

minimum value as in (7.21). For this relaxed procedure STRESS never increases, but we were not able to confirm the convergence result of ρ_m and η_m^2.

Note that for $q > 2$ zero distances are not replaced by ε in $A_s(\mathbf{X})$. This is valid since for $q > 2$, Hölder's inequality (7.8) has opposite sign, and simply states that $d_{ij}^2(\mathbf{X}) \geq 0$, which is clearly true.

7.3.2 Minkowski distances with $q < 1$

There is no metric interpretation for $d_{ij}(\mathbf{X})$ when $q < 1$, since the triangle inequality no longer holds, but the method can be applied along the same lines. It could be noted that the qth power of $d_{ij}(\mathbf{X})$ with $q < 1$ is metric (Carroll and Wish, 1974), but these so-called "extended" Minkowski-metrics are not considered here. For $q < 1$ the role of $\eta^2(\mathbf{X})$ and $\rho(\mathbf{X})$ is reversed. We can majorize STRESS by

$$\sigma^2(\mathbf{X}) \leq \eta_\delta^2 + \hat{\eta}^2(\mathbf{X},\mathbf{Y}) - 2\rho(\mathbf{X}) . \tag{7.27}$$

The Minkowski distance for $q < 1$ is a concave function in $\mathbf{X}$, which makes $-\rho(\mathbf{X})$ convex. Being the sum of two convex functions, the majorizing function (7.27) is also convex. The partial majorization algorithm for $q < 1$ consists of two steps: (1.) compute the majorizing function (7.27), (2.) compute the minimum of (7.27) for fixed $\mathbf{Y}$ and return to 1 unless convergence occurred. The solution of Step 2 is a global minimum since the right hand side of (7.27) is a convex function, although this need not be a unique solution. It can be computed by using a standard (convex) optimization technique. Furthermore, it can be proved that (7.27) is differentiable at a global minimum.

It seems much harder to derive convergence results on the sequence of η_m^2 and ρ_m. The sequence of σ_m^2 converges trivially because of majorization.

7.4 Differentiability at a local minimum

For Euclidean distances, De Leeuw (1984) proved that STRESS is always differentiable at a local minimum for *usable* data. He calls data usable if $w_{ij}\delta_{ij} > 0$ holds for all pairs i,j. Here we extend this result to Minkowski distances with $q > 1$.

The proof follows the one of De Leeuw (1984) closely. Although STRESS is not differentiable at points $\mathbf{X}$ where some $d_{ij}(\mathbf{X})$ are zero, it has directional derivatives in all directions at any point. The directional derivative of STRESS at $\mathbf{X}$ in direction $\mathbf{Y}$ is defined by

$$\nabla\sigma^2(\mathbf{X};\mathbf{Y}) = \lim_{\varepsilon \downarrow 0} \frac{\sigma^2(\mathbf{X} + \varepsilon\mathbf{Y}) - \sigma^2(\mathbf{X})}{\varepsilon} . \tag{7.28}$$

The directional derivative in direction $\mathbf{y}$ of a function $f(\mathbf{x})$ with gradient $g(\mathbf{x})$ equals $g(\mathbf{x})'\mathbf{y}$ if $g(\mathbf{x})$ exists at $\mathbf{x}$. The directional derivatives of the (squared) Minkowski distances are given by

$$\nabla d_{ij}(\mathbf{X};\mathbf{Y}) = \begin{cases} \dfrac{\sum_{s=1}^{p}(y_{is}-y_{js})|x_{is}-x_{js}|^{(q-1)}\operatorname{sign}(x_{is}-x_{js})}{d_{ij}^{q-1}(\mathbf{X})}, & \text{if } d_{ij}(\mathbf{X})\neq 0, \\ d_{ij}(\mathbf{Y}), & \text{if } d_{ij}(\mathbf{X})=0, \end{cases}$$

$$\nabla d_{ij}^2(\mathbf{X};\mathbf{Y}) = \begin{cases} 2\dfrac{\sum_{s=1}^{p}(y_{is}-y_{js})|x_{is}-x_{js}|^{(q-1)}\operatorname{sign}(x_{is}-x_{js})}{d_{ij}^{q-2}(\mathbf{X})}, & \text{if } d_{ij}(\mathbf{X})\neq 0, \\ 0, & \text{if } d_{ij}(\mathbf{X})=0. \end{cases} \tag{7.29}$$

For fixed $\mathbf{X}$, let $P = \{(i,j) \mid i<j,\ d_{ij}(\mathbf{X}) > 0\}$ and $Q = \{(i,j) \mid i<j,\ d_{ij}(\mathbf{X}) = 0\}$. Then the directional derivative of STRESS can be written as

$$\begin{aligned} \nabla\sigma^2(\mathbf{X};\mathbf{Y}) &= \textstyle\sum_{i<j} w_{ij}\nabla d_{ij}^2(\mathbf{X};\mathbf{Y}) - 2\sum_{i<j} w_{ij}\delta_{ij}\nabla d_{ij}(\mathbf{X};\mathbf{Y}) \\ &= 2\sum_{(i,j)\in P} \frac{w_{ij}\sum_{s=1}^{p}(y_{is}-y_{js})|x_{is}-x_{js}|^{(q-1)}\operatorname{sign}(x_{is}-x_{js})}{d_{ij}^{q-2}(\mathbf{X})} \\ &\quad -2\sum_{(i,j)\in P} \frac{w_{ij}\delta_{ij}\sum_{s=1}^{p}(y_{is}-y_{js})|x_{is}-x_{js}|^{(q-1)}\operatorname{sign}(x_{is}-x_{js})}{d_{ij}^{q-1}(\mathbf{X})} \\ &\quad -2\sum_{(i,j)\in Q} w_{ij}\delta_{ij}d_{ij}(\mathbf{Y}). \end{aligned} \tag{7.30}$$

If $\mathbf{X}$ is a local minimum, the directional derivative in all directions is nonnegative, which implies that for any $\mathbf{Y}$ we must have

$$\nabla\sigma^2(\mathbf{X};\mathbf{Y}) + \nabla\sigma^2(\mathbf{X};-\mathbf{Y}) = -4\sum_{(i,j)\in Q} w_{ij}\delta_{ij}d_{ij}(\mathbf{Y}) \geq 0. \tag{7.31}$$

Now choose $\mathbf{Y}$ such that $d_{ij}(\mathbf{Y}) > 0$ for all i,j. Whenever $w_{ij}\delta_{ij} > 0$ it follows that $d_{ij}(\mathbf{X}) > 0$. If $w_{ij}\delta_{ij} > 0$ for all i,j then $Q = \varnothing$. Hence, for usable data the STRESS function with Minkowski distances $q \geq 1$ has $d_{ij}(\mathbf{X}) > 0$ for all i,j. A zero distance can occur at a local minimum if and only if $w_{ij}\delta_{ij} = 0$. Moreover, if $q > 1$, or $p = 1$, this yields differentiability at each local minimum $\mathbf{X}$.

7.5 An example of MDS with Minkowski distances

To give an illustration of our algorithm we present a small example. Green, Carmone, and Smith (1989) reported preferences of 38 students for 10 varieties of cola. Every pair of

colas was judged on their similarity on a nine point rating scale. The dissimilarities were accumulated over the subjects and the result is reported in Table 7.1.

Table 7.1 *The dissimilarities between 10 colas reported by Green et al. (1989) accumulated over 38 judges.*

	Pepsi	Coke	Classic Coke	Diet Pepsi	Diet Slice	Diet 7-Up	Dr. Pepper	Slice	7-Up
Coke	127								
Classic Coke	169	143							
Diet Pepsi	204	235	243						
Diet Slice	309	318	326	285					
Diet 7-Up	320	322	327	288	155				
Dr. Pepper	286	256	258	259	312	306			
Slice	317	318	318	312	131	164	300		
7-Up	321	318	318	317	170	136	295	132	
Tab	238	231	242	194	285	281	256	291	297

Without loss of generality these values are normalized to have $\eta_\delta^2 = 1$. The cola data were analyzed in two dimensions with four different values of q, i.e., 1, 1.33, 1.66 and 2. For each value of q we used 25 different starting configurations. The iterations stopped whenever STRESS changed less than 10^{-8} in two subsequent iterations. Furthermore, we repeated the experiment using the relaxed update (that is, with $\alpha = 2$). The results of both experiments are given in Table 7.2. The third column in the table presents the best STRESS values given q. The best fit was obtained for $q = 1.33$. Inspection of the configuration corresponding to the best STRESS value suggested a diet non-diet dimension and a cola non-cola dimension. However, there was an inconsistency of the position of Diet 7-up on the diet non-diet dimension. Therefore, we positioned Diet 7-up in between Diet Slice and Diet Pepsi on the first dimension and restarted the algorithm. After 272 iterations a configuration was reached with lower STRESS value 0.03175500. This configuration is given in Figure 7.1. We deliberately do not display a two-dimensional representation, since it may lead to interpretations based on Euclidean distances. The first dimension appears to be a diet non-diet dimension. There is a large difference between Dr. Pepper and Tab, which is absent in the second dimension. The other beverages are separated in the second dimension into the group Pepsi, Coke, Classic Coke and Diet Pepsi and the other group of (Diet) Slice and (Diet) 7-Up. This dimension could be interpreted as a cola non-cola dimension.

Table 7.2 *The results of MDS of the cola data of Green et al. (1989) for different values of q.*

	Average STRESS	Average no of iterations	Lowest STRESS $\sigma^2(\mathbf{X}^*)$	No of iterations of $\sigma^2(\mathbf{X}^*)$
normal update				
$q = 1$	0.11713930	233.36	0.04785617	696
$q = 1.33$	0.05160195	251.96	0.03199579	284
$q = 1.66$	0.04086771	360.44	0.03491206	758
$q = 2$	0.04145104	145.92	0.03678052	141
relaxed update				
$q = 1$	0.11787413	100.40	0.04193646	232
$q = 1.33$	0.05483608	149.60	0.03425142	178
$q = 1.66$	0.04111878	243.32	0.03467676	350
$q = 2$	0.04070994	92.08	0.03685458	95
KYST				
$q = 1$	0.09727377	26.12	0.03697749	33
$q = 1.33$	0.05411852	29.60	0.03222712	26
$q = 1.66$	0.04333000	45.16	0.03469871	25
$q = 2$	0.04113443	38.68	0.03685562	39

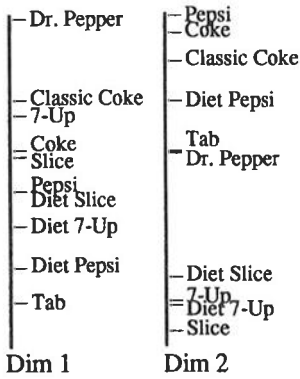


Figure 7.1 *The configuration of the best solution of the cola data, with q = 1.33 and having STRESS 0.03175500 .*

The second block in Table 7.2 suggests that the relaxed update needs less iterations than the normal update, as was expected. The average STRESS of 25 runs was generally larger for the relaxed update than the normal update, except for $q = 2$.

We also applied the KYST-2 program of Kruskal, Young, and Seery (1977) to this example. We specified Kruskal's STRESS formula 1 using linear regression without an intercept. This specification implies that up to a multiplicative constant β local minima coincide for both criteria. For the values reported in the third block of Table 7.2, we choose the optimum β, so that the STRESS values between the two methods are comparable. We see that the average STRESS values are similar for both algorithms, while the number of iterations is significantly smaller for KYST. Of course, this depends on the stopping criteria, which are weaker and more complicated to control in KYST. The lowest STRESS values of the majorization algorithm with and without relaxed update are generally lower, except in the case $q = 1$. Of course, the reason for our algorithm to find better STRESS values might be due to the difference of the stopping criteria. In fact, it is our experience that under standard choice of stopping criteria KYST frequently stops, even though the individual values of the gradient are significantly different from zero. Although nonzero gradients values also happens for the majorization algorithm with $q = 1$, the average squared sum of gradient values was roughly a factor 1000 smaller for the other q values. This phenomenon also accounts for the fact that the number of iteration is smaller for KYST. In summary, we conclude from this example that both algorithms perform comparably, with a slight advantage for the majorization algorithm in the intermediate q-range.

The differences between the average and minimum value of $\sigma^2(\mathbf{X})$ in the example show that the local minimum problem is quite severe, especially for q near to 1. Hubert, Arabie, and Hesson-McInnis (1992) seem to obtain better results by a combinatorial approach for the special case of the city-block metric in two dimensions.

7.6 Concluding remarks

In this chapter, we discussed the extension of the majorization approach to MDS using Minkowski distances. We presented an algorithm for Minkowski distances with $1 \leq q \leq 2$ that reduces STRESS at every iteration. For q outside this range an algorithm based on majorization was proposed that uses an inner optimization step in every iteration. Apart from a converging sequence of STRESS values, we could also prove some convergence properties of the separate components of the STRESS function. Further, we proved that at a local minimum the STRESS function with $q > 1$ is differentiable if all weights and dissimilarities are non-zero, which is an extension of the proof of De Leeuw (1984).

The current algorithm can be easily extended to deal with non-metric MDS. In that case, the procedure can be incorporated in an alternating least squares algorithm where the majorization step finds a better configuration and the optimal scaling step yields optimal

pseudo-distances, given the distances. For details we refer to Kruskal (1977) and De Leeuw and Heiser (1977).

The current algorithm gives an alternative for the approach that accommodates the possibility of negative dissimilarities discussed by Heiser (1991). Negative dissimilarities occur (among other situations) when city-block distances are fitted dimensionwise, as was done in Heiser (1989).

Some open questions with regard to the current algorithm remain. We have an indication that the local minimum problem is dependent on q and is especially severe for $q = 1$. Another interesting topic is how to incorporate constraints on the configuration (cf. De Leeuw and Heiser (1980), or section 1.7) in the majorization algorithm for STRESS with general Minkowski distance. These issues remain to be investigated.

APPENDIX

NOTATION

For convenience we summarize the notation used throughout this monograph. We use the following conventions: a lower case italic character denotes a scalar, a lower case bold character denotes a vector, and a uppercase bold character denotes a matrix. Elements of vectors or matrices are denoted by a subscripted scalar. A function is usually denoted by a character followed by an argument in parentheses, e.g., $f(\mathbf{x})$ is a scalar function of the vector $\mathbf{x}$, and $\mathbf{A}(\mathbf{x})$ is a matrix function of the vector $\mathbf{x}$. Some explicit notation follows below.

n	Number of objects.
i,j	Running index for objects, $i,j = 1,...,n$.
p	Number of dimensions.
s	Running index for dimensions, $s = 1,...,p$.
$\mathbf{X}$	Matrix of coordinates x_{ij} of n objects in p dimensions.
$\mathbf{X}^*$	A matrix of coordinates belonging to a local minimum.
δ_{ij}	Nonnegative dissimilarity between object i and j.
$\boldsymbol{\delta}$	Vector of all dissimilarities, of order $n(n-1)/2$.
$\boldsymbol{\Delta}$	Symmetric matrix of nonnegative dissimilarities δ_{ij} of size $n \times n$, with $\delta_{ii} = 0$.
$d_{ij}(\mathbf{X})$	Usually the Euclidean distance between row i and row j of $\mathbf{X}$, i.e. $d_{ij}^2(\mathbf{X}) = \sum_{s=1}^{p} (x_{is} - x_{js})^2$, except in chapter 7, where it denotes the Minkowski distance with $d_{ij}^q(\mathbf{X}) = \sum_{s=1}^{p} \lvert x_{is} - x_{js} \rvert^q$.
$\mathbf{d}$	Vector of all Euclidean distances between the rows of $\mathbf{X}$, of order $n(n-1)/2$.
$\mathbf{d}^*$	Vector of all Euclidean distances between the rows of $\mathbf{X}^*$, of order $n(n-1)/2$.
w_{ij}	A nonnegative weight used to (down)weight the residual in the STRESS function.
$\mathbf{W}$	Symmetric matrix of weights w_{ij} with zero diagonal.
$\mathbf{D}(\mathbf{X})$	Matrix of Euclidean distances between the rows of $\mathbf{X}$.
$\mathbf{B}(\mathbf{C};\mathbf{X})$	Matrix with off diagonal elements $b_{ij} = -c_{ij}/d_{ij}(\mathbf{X})$ if $d_{ij}(\mathbf{X}) \neq 0$ and $b_{ij} = 0$ otherwise and diagonal elements $b_{ii} = -\sum_{j \neq i} b_{ij}$.
$\mathbf{B}(\mathbf{X})$	Short form of the matrix $\mathbf{B}(\boldsymbol{\Delta};\mathbf{X})$.
$\mathbf{A}'$	The transpose of $\mathbf{A}$.
$\mathbf{A}^{-1}$	The inverse of a square matrix $\mathbf{A}$ assuming that $\mathbf{A}$ is of full rank, so that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$.

$\mathbf{A}^-$	A generalized inverse of a square matrix $\mathbf{A}$ where $\mathbf{A}$ may be rank deficient, for which we usually take the Moore-Penrose inverse so that $\mathbf{A}\mathbf{A}^-\mathbf{A} = \mathbf{A}$ and $\mathbf{A}^-\mathbf{A}\mathbf{A}^- = \mathbf{A}^-$ holds.
$\mathbf{A}(\mathbf{X})$	Any matrix function of $\mathbf{X}$.
tr $\mathbf{A}$	The trace operator sums the diagonal elements of $\mathbf{A}$, i.e. tr $\mathbf{A} = \sum_i a_{ii}$.
diag($\mathbf{a}$)	A diagonal matrix with diagonal elements equal to the elements of vector $\mathbf{a}$.
$\hat{\varphi}(\mathbf{x},\mathbf{y})$	Majorizing function of $\varphi(\mathbf{x})$ for which $\varphi(\mathbf{x}) \leq \hat{\varphi}(\mathbf{x},\mathbf{y})$ and $\varphi(\mathbf{y}) = \hat{\varphi}(\mathbf{y},\mathbf{y})$ holds for all feasible $\mathbf{x}$ and $\mathbf{y}$.
$\sigma^2(\mathbf{X})$	The square of the STRESS function, i.e. $\sigma^2(\mathbf{X}) = \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(\mathbf{X}))^2$.
A	Denotes a (possibly empty) set.
m(A)	The Lebesque measure gives a high dimensional volume measure of set A.
$\|\mathbf{X}\|$	The Euclidean norm of matrix $\mathbf{X}$, i.e. $\|\mathbf{X}\|^2 = \sum_{i,j}^{n} x_{ij}^2$.
$\|\mathbf{X}\|_\mathbf{V}$	The weighted Euclidean norm of matrix $\mathbf{X}$, i.e. $\|\mathbf{X}\|_\mathbf{V}^2 = \text{tr } \mathbf{X}'\mathbf{V}\mathbf{X}$.

REFERENCES

Arabie, P. (1991). Was Euclid an unnecessarily sophisticated psychologist? *Psychometrika, 56*, 567-587.

Bailey, R.A. & Gower, J.C. (1990). Approximating a symmetric matrix. *Psychometrika, 55*, 665-675.

Bezdek, J.C. (1981). *Pattern recognition with fuzzy objective function algorithms.* New York: Plenum Press.

Bijleveld, C.C.J.H. & De Leeuw, J. (1991). Fitting longitudinal reduced-rank regression models by alternating least squares. *Psychometrika, 56*, 433-447.

Boender, C.G.E. (1984). *The generalized multinomial distribution: A Bayesian analysis and applications* [Doctoral dissertation]. Rotterdam: Erasmus University.

Borg, I. & Lingoes, J. (1987). *Multidimensional similarity structure analysis.* New York: Springer-Verlag.

Carroll, J.D. & Chang, J.J. (1970). Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition. *Psychometrika, 35*, 439-463.

Carroll, J.D. & Wish, M. (1974). Multidimensional perceptual models and measurement methods. In: E.C. Carterette & M.P. Friedman (Eds.), *Handbook of perception, Vol. II*, 391-447. New York: Academic Press.

Černý, C. (1985). Thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications, 45*, 41-51.

Commandeur, J.J.F. (1991). *Matching configurations.* Leiden: DSWO Press, University of Leiden.

Coxon, A.P.M. (1982). *The user's guide to multidimensional scaling.* London: Heinemann.

De Amorim, G., Barthélemy, J.-P. & Ribeiro, C.C. (1992). Clustering and clique partitioning: Simulated annealing and tabu search approaches. *Journal of Classification, 9*, 17-41.

De Leeuw, J. (1977). Applications of convex analysis to multidimensional scaling. In: J.R. Barra, F. Brodeau, G. Romier & B. van Cutsem (Eds.), *Recent developments in statistics*, 133-145. Amsterdam: North-Holland.

De Leeuw, J. (1984). Differentiability of Kruskal's Stress at a local minimum. *Psychometrika, 49*, 111-113.

De Leeuw, J. (1988). Convergence of the majorization method for multidimensional scaling. *Journal of Classification, 5*, 163-180.

De Leeuw, J. (1992). Fitting distances by least squares. *Unpublished report.*

De Leeuw, J. & Heiser, W.J. (1977). Convergence of correction matrix algorithms for multidimensional scaling. In: J.C. Lingoes, E. Roskam & I. Borg (Eds.), *Geometric representations of relational data*, 735-752. Ann Arbor: Mathesis Press.

De Leeuw, J. & Heiser, W.J. (1980). Multidimensional scaling with restrictions on the configuration. In: P.R. Krishnaiah (Ed.), *Multivariate analysis, Vol. V*, 501-522. Amsterdam: North-Holland.

De Soete, G., Hubert, L. & Arabie, P. (1988). On the use of simulated annealing for combinatorial data analysis. In: W. Gaul and M. Schader (Eds.), *Data, expert, knowledge and decisions*, 329-340. Berlin: Springer-Verlag.

Defays, D. (1978). A short note on a method of seriation. *British Journal of Mathematical and Statistical Psychology, 3*, 49-53.

Dinkelbach, W. (1967). On nonlinear fractional programming. *Management Science, 13*, 492-498.

Dykstra, R.L. (1983). An algorithm for restricted least squares regression. *Journal of the American Statistical Association, 78*, 837-842.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7*, 179-188.

Fletcher, R. (1987). *Practical methods of optimization*. Chichester: Wiley.

Gaffke, N. & Mathar, R. (1989). A cyclic projection algorithm via duality. *Metrika, 36*, 29-54.

Gill, P.E., Murray, W. & Wright, M.H. (1981). *Practical optimization*. London: Academic Press.

Glover, F. (1989). Tabu search - Part I. *ORSA Journal on Computing, 1*, 190-206.

Glover, F. (1990). Tabu search - Part II. *ORSA Journal on Computing, 2*, 4-32.

Gomez, S. & Levy, A.V. (1982). The tunneling method for solving the constrained global optimization problem with non-connected feasible regions. In: J.P. Hennart (Ed.), *Lecture notes in mathematics, 909*, 34-47. Berlin: Springer-Verlag.

Gower, J.C. (1966). Some distance properties of latent roots and vector methods used in multivariate analysis. *Biometrika, 53*, 325-338.

Gower, J.C. (1975). Generalized Procrustes analysis. *Psychometrika, 40*, 33-51.

Gower, J.C. (1980). A modified Leverrier-Faddeev algorithm for matrices with multiple eigenvalues. *Linear Algebra and its Applications, 31*, 61-70.

Gower, J.C. (1984). Multivariate analysis: Ordination, multidimensional scaling and allied topics. In: E.H. Lloyd (Ed.), *Handbook of applicable mathematics, VI*, 727-781. Chichester: Wiley.

Gower, J.C. (1991). Personal communication.

Gower, J.C. & Groenen, P.J.F. (1991). Applications of the Modified Leverrier-Faddeev algorithm for the construction of explicit matrix spectral decompositions and inverses. *Utilitas Mathematica, 40*, 51-64.

Gower, J.C. & Legendre, P. (1986). Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification, 3*, 5-48.

Green, P.E., Carmone, F.J., Jr. & Smith, S.M. (1989). *Multidimensional scaling: Concepts and applications*. Boston: Allyn and Bacon.

Groenen, P.J.F. (1990). The tunneling method applied to multidimensional scaling: Progress report 1. *Research Report RR-90-03*. Leiden: Department of Data Theory.

Groenen, P.J.F. (1992). A comparison of two methods for global optimization in multidimensional scaling. In: O. Opitz, B. Lausen & R. Klar (Eds.), *Information and classification: Concepts, methods and applications*, 145-155. Berlin: Springer-Verlag.

Groenen, P.J.F. (1993). An application of a Modified Leverrier-Faddeev algorithm for partitioned block designs in multidimensional scaling. In: R. Steyer, K.F. Wender & K.F. Widaman (Eds.), *Psychometric methodology: Proceedings of the 7th European meeting of the Psychometric Society in Trier*, 151-156. Stuttgart: Gustav Fischer Verlag.

Groenen, P.J.F. & Heiser, W.J. (1991). An improved tunneling function for finding a decreasing series of local minima. *Research Report RR-91-06*. Leiden: Department of Data Theory.

Groenen, P.J.F. & Heiser, W.J. (1992). *A study of acceleration schemes for least squares scaling using semi-complete MDS*. Paper presented at the Distancia '92 conference, Rennes.

Groenen, P.J.F., Mathar, R. & Heiser, W.J. (1992). The majorization approach to multidimensional scaling for Minkowski distances. *Research Report RR-92-11*. Leiden: Department of Data Theory.

Guttman, L. (1968). A general nonmetric technique for finding the smallest coordinate space for a configuration of points. *Psychometrika, 33*, 469-506.

Hansen, E. (1980). Global optimization using interval analysis: The multidimensional case. *Numerische Mathematik, 34*, 247-270.

Hardy, G.H., Littlewood, J.E. & Pólya, G. (1952). *Inequalities* (2nd edition, 1952). Cambridge: University Press.

Hartigan, J.A. (1975). *Clustering algorithms*. New York: Wiley.

Heiser, W.J. (1981). *Unfolding analysis of proximity data* [Doctoral dissertation]. Leiden: University of Leiden.

Heiser, W.J. (1987). Joint ordination of species and sites: The unfolding technique. In: P. Legendre & L. Legendre (Eds.). *Developments in numerical ecology*, 189-221. Berlin: Springer-Verlag.

Heiser, W.J. (1988). Multidimensional scaling with least absolute residuals. In: H.H. Bock (Ed.), *Classification and related methods of data analysis*, 455-462. Amsterdam: North-Holland.

Heiser, W.J. (1989). The city-block model for three-way multidimensional scaling. In: R. Coppi and S. Bolasco (Eds.), *Multiway data analysis*, 395-404. Amsterdam: North-Holland.

Heiser, W.J. (1991). A generalized majorization method for least squares multidimensional scaling of pseudodistances that may be negative. *Psychometrika, 56*, 7-27.

Heiser, W.J. (1992). Clustering in low-dimensional space. In: O. Opitz, B. Lausen & R. Klar (Eds.), *Information and classification: Concepts, methods and applications*, 145-155. Berlin: Springer-Verlag.

Heiser, W.J. & De Leeuw, J. (1977). How to use SMACOF-I (3rd edition, 1986). *Research Report UG-86-02*. Leiden: Department of Data Theory.

Horst, R. & Tuy, H. (1990). *Global optimization: Deterministic approaches*. Berlin: Springer-Verlag.

Hubert, L.J. & Arabie, P. (1986). Unidimensional scaling and combinatorial optimization. In: J. De Leeuw, W.J. Heiser, J. Meulman & F. Critchley (Eds.), *Multidimensional data analysis*, 181-196. Leiden: DSWO Press, University of Leiden.

Hubert, L.J., Arabie, P. & Hesson-McInnis, M. (1992). Multidimensional scaling in the city-block metric: A combinatorial approach. *Journal of Classification, 9*, 211-236.

Hubert, L.J. & Golledge, R.G. (1981). Matrix reorganization and dynamic programming: Applications to paired comparisons and unidimensional seriation. *Psychometrika, 46*, 429-441.

Ihm, P. (1986). A problem on bacterial taxonomy. *Working paper*. EURATOM: Ispra.

Kiers, H.A.L. (1990). Majorization as a tool for optimizing a class of matrix functions. *Psychometrika, 55*, 417-428.

Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science, 220*, 671-680.

Klaassen, P. (1989). *Globale optimalisatie & multidimensional scaling* [Unpublished master's thesis]. Rotterdam: Faculty of Economics, Erasmus University.

Kruskal, J.B. (1964a). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika, 29*, 1-28.

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

Kruskal, J.B. (1977). Multidimensional scaling and other methods for discovering structure. In: K. Enslein, A. Ralston & H.S. Wilf (Eds.), *Statistical methods for digital computers, Vol. III*, 296-339. New York: Wiley.

Kruskal, J.B. & Wish, M. (1978). Multidimensional scaling. *Sage University Paper Series on Quantitative Applications in the Social Sciences, series no. 07-011.* Beverly Hills and London: Sage Publications.

Kruskal, J.B., Young, F.W. & Seery, J.B. (1977). *How to use KYST-2, a very flexible program to do multidimensional scaling and unfolding.* Murray Hill, NJ: AT&T Bell Laboratories.

Levy, A.V. & Gomez, S. (1985). The tunneling method applied to global optimization. In: P.T. Boggs, R.H. Byrd & R.B. Schnabel (Eds.), *Numerical optimization 1984*, 213-244. Philadelphia: SIAM.

Li, Y. & Pardalos, P.M. (1992). Generating quadratic assignment test problems with known optimal permutation. *Computational Optimization and Applications, 1*, 163-184.

Luenberger, D.G. (1973). *Introduction to linear and nonlinear programming.* Reading, MA: Addison-Wesley.

Machmouchi, M. (1992). *Contributions à la mise en œvre des méthodes d'Analyse des Données de Dissimilarité* [Doctoral dissertation]. Grenoble: University of Grenoble II.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In: *5th Berkeley Symposium on Mathematics, Statistics, and Probability, Vol. 1*, 281-298.

Magnus, J.R. & Neudecker, H. (1988). *Matrix differential calculus with applications in statistics and econometrics.* Chichester: Wiley.

Mathar, R. (1989). Algorithms in multidimensional scaling. In: O. Opitz (Ed.), *Conceptual and numerical analysis of data*, 159-177. Berlin: Springer-Verlag.

Mathar, R. (1991). Personal communication.

Mathar, R. & Groenen, P.J.F. (1991). Algorithms in convex analysis applied to multidimensional scaling. In: E. Diday and Y. Lechevallier (Eds.), *Symbolic-numeric data analysis and learning*, 45-56. Commack, NY: Nova Science Publishers.

Mathar, R. & Meyer, R. (1992). *Algorithms in convex analysis to fit l_p-distance matrices* [Submitted for publication]. Aachen: RWTH.

Meulman, J.J. (1986). *A distance approach to nonlinear multivariate analysis.* Leiden: DSWO Press, University of Leiden.

Meulman, J.J. (1992). The integration of multidimensional scaling and multivariate analysis with optimal transformations. *Psychometrika, 57*, 539-565.

Mockus, J. (1989). *Bayesian approach to global optimization.* Dordrecht: Kluwer Academic Publishers.

Montalvo, A. (1979). *Development of a new algorithm for the global minimization of functions* [Ph.D. thesis]. Mexico: Universidad Nacional Autonoma de Mexico.

Ortega, J.M. & Rheinboldt, W.C. (1970). *Iterative solutions of nonlinear equations in several variables.* New York: Academic Press.

Ostrowski, A.M. (1966). *Solution of equations and systems of equations.* New York: Academic Press.

Peters, G. & Wilkinson, J.H. (1971). The calculation of specified eigenvectors by inverse iteration. In: J.H. Wilkinson & C. Reinsch (Eds.), *Handbook for automatic computation, Vol. II, Linear algebra*, 418-439. Berlin: Springer-Verlag.

Poole, K.T. (1984). Least squares metric, unidimensional unfolding. *Psychometrika, 49*, 311-323.

Poole, K.T. (1990). Least squares metric, unidimensional scaling of multivariate linear models. *Psychometrika, 55*, 123-149.

Press, W.P., Flannery, B.P., Teukolsky, S.A. & Vetterling, W.T. (1988). *Numerical recipes.* New York: Cambridge University Press.

Ramsay, J.O. (1969). Some statistical considerations in multidimensional scaling. *Psychometrika, 34*, 167-182.

Ramsay, J.O. (1977). Maximum likelihood estimation in MDS. *Psychometrika, 42*, 241-266.

Rinnooy Kan, A.H.G. & Timmer, G.T. (1987a). Stochastic global optimization methods, part I: Clustering methods. *Mathematical Programming, 39*, 27-56.

Rinnooy Kan, A.H.G. & Timmer, G.T. (1987b). Stochastic global optimization methods, part II: Multi level methods. *Mathematical Programming, 39*, 57-78.

Robinson, W.S. (1951). A method for chronologically ordering archaeological deposits. *American Antiquity, 16*, 293-301.

Rockafellar, R.T. (1970). *Convex analysis*. Princeton: Princeton University Press.

Schiffman, S.S., Reynolds, M.L. & Young, F.W. (1981). *Introduction to multidimensional scaling*. London: Academic Press.

Shepard, R.N. (1962). Analysis of proximities: Multidimensional scaling with an unknown distance function. *Psychometrika, 27*, 125-140.

Spence, I. (1982). Incomplete experimental designs for multidimensional scaling. In: R.G. Golledge & J.N. Rayner (Eds.), *Proximity and preference: Problems in the multidimensional analysis of large data sets*, 29-46. Minneapolis: University of Minnesota Press.

Spence, I. & Domoney, D.W. (1974). Single subject incomplete designs for nonmetric multidimensional scaling. *Psychometrika, 39*, 469-490.

Stoop, I. & De Leeuw, J. (1982). How to use SMACOF-1B. *Research Report*. Leiden: Department of Data Theory.

Takane, Y., Young, F.W., & De Leeuw, J. (1977). Nonmetric individual differences in multidimensional scaling: An alternating least squares method with optimal scaling features. *Psychometrika, 42*, 7-67.

Tijssen, R.J.W. (1992). *Cartography of science: Scientometric mapping with multidimensional scaling methods*. Leiden: DSWO Press, University of Leiden.

Timmer, G.T. (1984). *Global optimization: A stochastic approach* [Doctoral dissertation]. Rotterdam: Erasmus University.

Torgerson, W.S. (1958). *Theory and methods of scaling*. New York: Wiley.

Törn, A.A. (1976). Cluster analysis using seed points and density determined hyperspheres with an application to global optimization. In: *Proceedings of the third international conference on pattern recognition*. Coronado, California.

Törn, A.A. & Žilinskas, A. (1989). Global optimization. In: G. Goos & J. Hartmanis (Eds.), *Lecture notes in computer science, Vol. 350*. Berlin: Springer-Verlag.

Van der Lans, I.A. (1992). *Nonlinear multivariate analysis for multiattribute preference data*. Leiden: DSWO Press, University of Leiden.

Van Laarhoven, P.J.M. & Aarts, E.H.L. (1987). *Simulated annealing: Theory and applications*. Dordrecht: Kluwer Academic Publishers.

Verboon, P. & Heiser, W.J. (1992). Resistant orthogonal Procrustes analysis. *Journal of Classification, 9*, 237-256.

Vilkov, A.V., Zhidkov, N.P. & Shchedrin, B.M. (1975). A method of finding the global minimum of a function of one variable. *USSR Computational Mathematics and Mathematical Physics, 15*, 1040-1042.

Wagenaar, W.A. & Padmos, P. (1971). Quantitative interpretation of stress in Kruskal's method multidimensional scaling technique. *British Journal of Mathematical and Statistical Psychology, 24*, 101-110.

Wilkinson, J.H. (1965). *The algebraic eigenvalue problem*. Oxford: University Press.

Zangwill, W.I. (1969). *Nonlinear programming, a unified approach*. Englewood Cliffs, NJ: Prentice Hall.

Zhigljavsky, A.A. (1991). *Theory of global random search*. Dordrecht: Kluwer Academic Publishers.

Zielman, B. (1991). Three-way scaling of asymmetric proximities. *Research Report RR-91-01*. Leiden: Department of Data Theory.

Žilinskas, A. (1978). Optimization of one-dimensional multimodal functions, Algorithm AS 133. *Applied Statistics*, *23*, 367-375.

AUTHOR INDEX

SUBJECT INDEX

SUMMARY

This monograph is concerned with technical aspects of multidimensional scaling (MDS). MDS is a technique that aims at representing objects as points in a low dimensional space such that the distance between each pair of points matches the interobject dissimilarities as closely as possible. Usually, the dissimilarities –a measure indicating how dissimilar each pair of objects is– are gathered directly or derived indirectly. In this monograph we assume that the dissimilarities are given. One way to do MDS is to minimize the sum of squared differences of dissimilarities and distances over the coordinates of the points. This function is called Kruskal's STRESS and is used throughout this monograph. We minimize STRESS using iterative majorization, which guarantees that the STRESS values become lower, or remain the same. However, some issues remain open. While majorization guarantees lower (or equal) STRESS values and almost always stops at a local minimum, the minimum need not be the overall best local minimum, the global minimum. This problem is especially severe for unidimensional scaling, where the points are forced to be on a line. Therefore, an important part of this monograph is devoted to methods that aim at finding the global minimum of STRESS. One such method, the tunneling method, is discussed in great detail in chapter 3. In chapter 4 we try to find indications when local minima may be found, and get an idea of the performance of some global optimization methods. Another aspect discussed in this monograph is the inclusion of weights in the STRESS function. It allows for missing patterns with multidimensional scaling. We discuss several structured missing data patterns, for which some computations can be accelerated. Also, we indicate how missing data patterns can be used for (external) unfolding and semi-complete scaling. To facilitate the interpretation of multidimensional scaling solutions with a large number of objects, we elaborate on a method called cluster differences scaling that does clustering and MDS simultaneously. The final topic in this monograph is concerned with the extension of the majorization method for STRESS with Minkowski distances. We continue with a more detailed overview of the chapters.

In the first chapter we give a brief introduction to multidimensional scaling. We discuss several aspects of iterative majorization, which is the prime minimization method used in this monograph. Majorization is used to approximate a complicated function by a more simple auxiliary function, the majorizing function. This function touches the complicated function at the current estimate, the suppporting point, and is located above it everywhere else. At the minimum of the majorizing function, the original function value must be lower than or equal to the value at the supporting point, since the majorizing function is by definition located above the original function. This point becomes the supporting point for the next majorization function. Iterating this process gives

increasingly better estimates and nearly always leads to a local minimum. Two useful types of majorizing functions are distinguished: linear majorization and quadratic majorization. Then we show how majorization is used to obtain a convergent algorithm for minimizing STRESS, give its rate of convergence, and present an algorithm based on distances only. Although the majorization algorithm usually leads to a local minimum, it need not be the global minimum. We split the problem into three cases: unidimensional scaling, multidimensional scaling, and fulldimensional scaling. We show that unidimensional scaling is a combinatorial problem with many local minima. For fulldimensional scaling we prove that STRESS only has local minima that are global minima. For multidimensional scaling we may have multiple local minima.

Chapter 2 starts with a classification of global optimization techniques. Some of them are discussed in more detail and applied to the STRESS function. The combinatorial problem of unidimensional scaling can be solved globally by dynamic programming, or solved locally by using heuristic strategies like simulated annealing, the tabu search, and pairwise interchange strategies, of which four are discussed. For multidimensional scaling we discuss several techniques, like space covering methods, multistart, and multi-level-single-linkage clustering. We adapt the multi-level-single-linkage clustering algorithm to make it usable with STRESS. We prove that STRESS has a Lipschitz constant.

In chapter 3 the tunneling method for finding a global minimum is presented. It aims at finding an ever decreasing series of local minima. The method alternates between a local search and a tunneling step in which a configuration is sought different from the previous local minimum, but with equal or less STRESS than the previous local minimum. The latter step is performed by minimizing the tunneling function, which is modelled such that trivial solutions are excluded and that it behaves well for the STRESS function. Furthermore, we present a minimization algorithm for the tunneling function based on majorization. An extension of the algorithm is given to accommodate more than one pole, which is needed to avoid unwanted stationary points of the tunneling function. The algorithm uses parametric programming, which is extended to deal with majorization. Several new majorization inequalities are presented. In two examples we show that the tunneling algorithm works, at least for these small problems. An experiment is presented to find proper tuning of the parameters that need to be set in the tunneling function. More numerical experiments of the tunneling method are given in chapter 4.

In the fourth chapter we study how the algorithms succeed in finding the global minimum. For multidimensional scaling, our experiment suggests that the number of local minima increases as the error increases, decreases as the number of objects increases, and decreases as the dimensionality gets higher. Numerical experiments indicate that the tunneling algorithm is equally capable in finding the candidate global minimum as is multistart or multi-level-single-linkage clustering. However, all these methods are computationally very demanding. The settings of the tunneling method (like maximum number of poles, maximum number of iterations in a tunneling step, convergence criterion of the tunneling function) need to be strong not to fail. For multi-

level-single-linkage clustering the use of moved hypercubes greatly reduces the number of local searches, and still recovers the same candidate global minimum in our experiment. For unidimensional scaling the first choice should be the dynamic programming approach if all weights are constant, since it guarantees a global optimum. For a large number of objects (say larger than 20), or with non-identical weights, the dynamic programming strategy is no longer feasible, and heuristic strategies, like pairwise interchange, are to be used. These local pairwise interchange strategies (LOPI) and the tabu search are compared on their capability in recovering a generated global optimum. In fact, the LOPI2 strategy (that compares the interchanges formed by inserting one object keeping the order of the other objects fixed) and the LOPI3 (that accepts any pairwise interchange with a better fit) give a good performance within a reasonable amount of CPU time. The LOPI1 (adjacent pairwise interchanges) and LOPI4 (that accepts the pairwise interchange of all with the best fit) strategies did not perform very favourably. The former is fast, but often does not locate the global optimum, whereas the latter is able to find the global optimum, but takes a huge amount of CPU time. The tabu search performed only a little bit better than LOPI1, but not as good as the other pairwise interchange strategies. A small example shows that the tunneling algorithm still works for unidimensional scaling, but is not successful in finding the global optimum.

In chapter 5 we discuss several computational aspects of incomplete MDS, where some of the weights for the pairs are set equal to zero. First, we treat three structured designs –partitioned block design, the block tridiagonal design, and the circular design– and show how the computation of the inverse needed for these designs can be accelerated dramatically. Then, we discuss a classification of models based on splitting the set of objects into two parts. Using this classification some known models (like (external) unfolding and complete MDS) and new models (cyclic point descent, semi-complete MDS, almost complete MDS) are distinguished. Finally, we discuss an application of semi-complete MDS aimed at accelerating the MDS algorithm. We call this moving frame MDS, because we iteratively select a frame of well fitting points, keep them fixed, and fit the other points using semi-complete MDS. Numerical experiments show that moving frame MDS is always much slower than complete MDS. The fastest computations are obtained for complete MDS using the relaxed update. It turns out that when using the relaxed update we always have to adjust the scaling size of the coordinates. Otherwise, the complete MDS algorithm may stop too early.

In chapter 6 we elaborate on cluster differences scaling (CDS) for MDS originally proposed by Heiser (1992). The aim is to facilitate interpretation and reduce the number of parameters by imposing cluster restrictions on the objects and still get a graphical representation of the clusters to model the dissimilarities. We indicate how the STRESS in CDS can be decomposed into four additive components of within and between cluster STRESS. We show that CDS can be seen as ordinary MDS with cluster restrictions on the configuration. Furthermore, we generalize CDS to deal with fuzzy clusters, which has CDS as a special case. By an experiment we show that a start configuration for CDS based on repeated fuzzy clustering with decreasing fuzzy cluster parameter q, gives the lowest

STRESS value. This is a particularly useful application of fuzzy CDS to avoid local minima in CDS.

In the last chapter we discuss the extension of the majorization approach to MDS using Minkowski distances, which has the city-block distance, the Euclidean distance, and the max distance as special cases. We present an algorithm for Minkowski distances between city-block and Euclidean distances that reduces STRESS at every iteration. For other Minkowski distances a majorization algorithm is proposed that uses an inner optimization step in every iteration. Apart from a converging sequence of STRESS values, we prove some convergence properties of the separate components of the STRESS function. Furthermore, for Minkowski distances with power parameter larger than city-block distances we prove that at a local minimum the STRESS function is differentiable if all weights and dissimilarities are positive. This new algorithm is compared with a standard algorithm, i.e., KYST. It seems that both algorithms behave similar, with a slight advantage for the majorization algorithm for some Minkowski distances.

THE MAJORIZATION APPROACH TO MULTIDIMENSIONAL SCALING

SOME PROBLEMS AND EXTENSIONS

Multidimensional scaling (MDS) methods fit a distance model to one or more tables of dissimilarities. Associated with the distance model is a spatial representation of the objects of analysis, the objective being that the interpoint distance in space should match the interobject dissimilarity in the table as closely as possible. Usually, the distance model is Euclidean, but in this monograph there is one chapter in which non-Euclidean metrics are considered as well.

Working in a least squares framework, a number of unsolved technical problems in this area are discussed and some of them resolved. The technical apparatus used is the iterative majorization approach, developed in the late seventies by De Leeuw and Heiser, supplemented here with several other state-of-the-art optimization methods. The most important topic discussed is the well-known problem that the MDS badness-of-fit function generally has multiple local minima, in which the usual search procedures may easily get trapped (especially in the one-dimensional case). To resolve this problem, an existing global search procedure, the tunneling method, is fully integrated in the iterative majorization framework, and compared with a stochastic method of global optimization. In terms of performance in unconstrained MDS, they are about equal, but the tunneling method is more flexible and could easily be adapted for constrained MDS models. Special methods for the one-dimensional case, such as dynamic programming and several pairwise interchange strategies, are also discussed and evaluated.

Apart from the global minimum problem, three other topics are covered in detail: efficient ways to perform different forms of incomplete MDS; the use of fuzzy memberships in cluster differences scaling, a form of MDS with cluster constraints; and the extension of the iterative majorization approach to fitting models with a Minkowski metric. In sum, this monograph covers the current state of algorithmic affairs in least squares MDS, but it should also be valuable for readers interested, more generally, in global and convergent optimization of objective functions with many variables.

DSWO PRESS

ISBN 90-6695-086-2