Global Minima by Penalized Full-dimensional Scaling

Jan de Leeuw

First created on May 06, 2019. Last update on June 30, 2023

Abstract

The full-dimensional (metric, Euclidean, least squares) multidimensional scaling stress loss function is combined with a quadratic external penalty function term. The trajectory of minimizers of stress for increasing values of the penalty parameter is then used to find (tentative) global minima for low-dimensional multidimensional scaling. This is illustrated with several one-dimensional and two-dimensional examples.

Contents

Note: This is a working paper which will be expanded/updated frequently. All suggestions for improvement are welcome. The directory deleeuwpdx.net/pubfolders/penalty has a pdf version, a html version, the bib files, the complete Rmd file with the code chunks, and the R source code.

Introduction

We start with a reformulation of the (metric, Euclidean, least squares) multidimensional scaling (MDS) problem. The main difference with previous formulations, described most thoroughly in Borg and Groenen (2005), is that we parametrize the problem by using the *cross-product matrix*, and not by using the *configuration matrix*.

The *data* of an MDS problem are non-negative, symmetric, and hollow matrices $W = \{w_{ij}\}$ and $\Delta = \{\delta_{ij}\}$ of order n, containing *weights* and *dissimilarities*. We assume that W is irreducible, so that the MDS problem is not equivalent to a number of smaller MDS problems.

For any $1 \leq p \leq n-1$ we define the *pMDS problem* as the minimization od

$$\sigma(C) = \frac{1}{2}\sum\sum_{1\leq i<j\leq n} w_{ij}(\delta_{ij} - d_{ij}(C))^2 \tag{1}$$

over all $n \times n$ matrices C which satisfy

1. C is symmetric and doubly-centered (SDC), i.e. rows and columns add up to zero.
2. $C \gtrsim 0$, i.e. C is positive semi-definite (PSD).
3. $\text{rank}(C) \leq p$.

Here $D(C) = \{d_{ij}(C)\}$ is a matrix of Euclidean distances, i.e. $d_{ij}(C) = \sqrt{c_{ii} + c_{jj} - 2c_{ij}}$. Special cases are unidimensional scaling, which is 1MDS, and full-dimensional scaling, which is (n-1)MDS. They deserve their own acronyms, so we'll refer to them as UDS and FDS.

We now introduce some standard MDS notation, following De Leeuw (1977). Define the unit vectors e_i, which have element i equal to one and all other elements equal to zero. For $i < j$ define the matrices

$$A_{ij} = (e_i - e_j)(e_i - e_j)'.$$

Note that $d_{ij}(C)=\sqrt{\text{tr }A_{ij}C}$. Also note the A_{ij} are a basis for the $\frac12 n(n-1)$ dimensional subspace of SDC matrices of order n. In fact for any SDC matrix $C=\{c_{ij}\}$ we have

$$C=-\sum\sum_{1\leq i<j\leq n}c_{ij}A_{ij}.$$

Next, define the matrix $V=\{v_{ij}\}$ by

$$V=\sum\sum_{1\leq i<j\leq n}w_{ij}A_{ij}. \tag{2}$$

Matrix V is PSD and SDC, and because of irreducibility it has rank $n-1$. The vectors in the null-space of V are all proportional to u, a vector with all elements equal to one.

Also define the matrix valued function $B(C)=\{b_{ij}(C)\}$ by

$$B(C)=\sum\sum_{1\leq i<j\leq n}w_{ij}r_{ij}(C)A_{ij} \tag{3}$$

where

$$r_{ij}(C)=\begin{cases}\frac{\delta_{ij}}{d_{ij}(C)} & \text{if } d_{ij}(C)>0,\\ 0 & \text{if } d_{ij}(C)=0.\end{cases}$$

q If C is PSD and SDC then $B(C)$ is also PSD and SDC. Its rank is less than or equal to $n-1$, with equality if and only if the Hadamard product $W\star R(C)$ is irreducible.

We also assume, without loss of generality, that dissimilarities are normalized as

$$\frac12\sum\sum_{1\leq i<j\leq n}w_{ij}\delta_{ij}^2=1.$$

With these definitions we can rewrite the stress (1) as

$$\sigma(C)=1-\textbf{tr }B(C)C+\frac12\textbf{tr }VC. \tag{4}$$

Configuration

Suppose $C=ZZ'$, with Z and $n\times p$ matrix of rank p. If σ has a local minimum at C then $(V-B(C))Z=0$ and $d_{ij}(C)>0$ for all $i<j$ with $w_{ij}\delta_{ij}>0$.

$C=ZZ'$ $\tilde C=(Z+\epsilon E)(Z+\epsilon E)'=C+\epsilon(EZ'+ZE')+\epsilon^2EE'$

$$d_{ij}^2(\tilde C)=d_{ij}^2(C)+2\epsilon\text{ tr }Z'A_{ij}E+\epsilon^2d_{ij}^2(F)$$

If $d_{ij}(C)=0$ then $d_{ij}^2(\tilde C)=\epsilon^2d_{ij}^2(F)$

If $d_{ij}(C)>0$

$$d_{ij}(\tilde C)=d_{ij}(C)\sqrt{1+2\epsilon\ \frac{\text{tr }Z'A_{ij}E}{d_{ij}^2(C)}+\epsilon^2\frac{d_{ij}^2(F)}{d_{ij}^2(C)}}$$

$$d_{ij}(\tilde{C}) = d_{ij}(C) + \epsilon\ \frac{\text{tr}\ Z'A_{ij}E}{d_{ij}(C)} + \frac{1}{2}\epsilon^2 \left\{\frac{d_{ij}^2(F)}{d_{ij}(C)} - \frac{(\text{tr}\ Z'A_{ij}E)^2}{d_{ij}^3(C)}\right\} + o(\epsilon^2)$$

$$\sigma(\tilde{C}) = \sigma(C) + \epsilon\ \text{tr}\ Z'(V_+ - B(C))E - \epsilon \sum_{d_{ij}(C)=0} w_{ij}\delta_{ij}d_{ij}(F) +$$
$$+\frac{1}{2}\epsilon^2 \left[\text{tr}\ VF - \sum_{d_{ij}(C)>0} w_{ij}r_{ij}(C)\left\{d_{ij}^2(F) - \frac{((z_i - z_j)'(e_i - e_j))^2}{d_{ij}^2(C)}\right\}\right] + o(\epsilon^2)$$

FDS

The definition (4) shows that the CFDS loss function is a convex function on the cone of positive semi-definite matrices, because the square root of a non-negative linear function of the elements of C is concave. Positivity of the weights and dissimilarities implies that loss is actually strictly convex. The necessary and sufficient conditions for C to be the unique solution of the CFDS problem are simply the conditions for a proper convex function to attain its minimum at C on a closed convex cone (Rockafellar (1970), theorem 31.4).

$$\begin{aligned} V - B(C) &\gtrsim 0, \\ C &\gtrsim 0, \\ \mathbf{tr}\ C(V - B(C)) &= 0. \end{aligned}$$

The conditions say that C and $V - B(C)$ must be positive semi-definite and have complimentary null spaces.

If C is the solution of the FMDS problem then $\mathbf{rank}(C)$ is called the *Gower rank* of the MDS problem defined by W and Δ (De Leeuw (2016)). Although there is a unique Gower rank associated with each pair (W, Δ), we can also talk about the *approximate Gower rank* by ignoring the small eigenvalues of C.

FDS using SMACOF

The usual SMACOF algorithm can be applied to FMDS as well. The iterations start with $Z^{(0)}$ and use the update rule

$$Z^{(k+1)} = V^+B(Z^{(k)})Z^{(k)}, \tag{5}$$

where V^+ is the Moore-Penrose inverse of V, and is consequently also doubly-centered. This means that all $Z^{(k)}$ in the SMACOF sequence, except possibly $Z^{(0)}$, are column-centered and of rank at most $n - 1$. Equation (5) also shows that if $Z^{(0)}$ is of rank $p < n - 1$ then all $Z^{(k)}$ are of rank p as well.

De Leeuw (1977) shows global convergence of the SMACOF sequence for pMDS, generated by (5), to a stationary point, i.e. a point satisfying $(V - B(Z))Z = 0$. This result also applies, of course, to nMDS, i.e. FDS. If Z is a solution of the stationary equations then with $C = ZZ'$ we have both $(V - B(C))C = 0$ and $C \gtrsim 0$, but since we generally do not have $V - B(Z) \gtrsim 0$, this does not mean that C solves the CFDS problem.

In fact, suppose the unique CMDS solution has Gower rank $r \geq 2$. Start the SMACOF FDS iterations (5) with $Z^{(0)}$ of the form $Z^{(0)} = \begin{bmatrix} X^{(0)} & \mid & 0 \end{bmatrix}$, where $X^{(0)}$ is an $n \times p$ matrix of rank $p < r$. All $Z^{(k)}$ will be of this form and will also be of rank p, and all accumulation points Z of the SMACOF sequence will have this form and $\mathbf{rank}(Z) \leq p$. Thus $C = ZZ'$ cannot be the solution of the CMDS problem.

The next result shows that things are allright, after all. Although stress in FDS is certainly not a convex function of Z, it remains true that all local minima are global.

Lemma 1: [Expand] If FDS stress has a local minimum at $\begin{bmatrix} X & \mid & 0 \end{bmatrix}$, where X is $n \times p$ and the zero block is $n \times q$ with $q > 1$, then

1: $\mathcal{D}\sigma(X) = (V - B(X))X = 0.$

2: $\mathcal{D}^2\sigma(X) \gtrsim 0.$

3: $V - B(X) \gtrsim 0.$

Proof: We use the fact that stress is differentiable at a local minimum (De Leeuw (1984)). If $Z = \begin{bmatrix} X & \mid & 0 \end{bmatrix} + \epsilon \begin{bmatrix} P & \mid & Q \end{bmatrix}$ then we must have $\sigma(Z) \geq \sigma(X)$ for all P and Q. Now

$$\sigma(Z) = \sigma(X) + \epsilon \text{ tr } P'\mathcal{D}\sigma(X) + \\ + \frac{1}{2}\epsilon^2\ \mathcal{D}^2\sigma(X)(P,P) + \frac{1}{2}\epsilon^2 \text{ tr } Q'(V - B(X))Q + o(\epsilon^2). \quad (6)$$

The lemma follows from this expansion. ■

Theorem 1: [FDS Local Minima] If stationary point Z of FDS is a local minimum, then it also is the global minimum, and $C = ZZ'$ solves the CFDS problem.

Proof: We start with a special case. Suppose Z is a doubly-centered solution of the FDS stationary equations with $\mathbf{rank}(Z) = n - 1$. Then $(V - B(Z))Z = 0$ implies $V = B(Z)$, which implies $\delta_{ij} = d_{ij}(Z)$ for all i, j. Thus $\sigma(Z) = 0$, which obviously is the global minimum.

Now suppose Z is a doubly-centered local minimum solution of the FDS stationary equations with $\mathbf{rank}(Z) = r < n - 1$. Without loss of generality we assume Z is of the form $Z = \begin{bmatrix} X & \mid & 0 \end{bmatrix}$, with X an $n \times r$ matrix of rank r. For $C = ZZ'$ to be a solution of the CFDS problem it is necessary and sufficient that $V - B(Z) \gtrsim 0$. Lemma 1 shows that this is indeed the case at a local minimum. ■

Corrollary 1: [Saddle] A pMDS solution of the stationary equations with Z singular is a saddle point.

Corrollary 2: [Nested] Solutions of the stationary equations of pMDS are saddle points of qMDS with $q > p$.

The proof of lemma 1 shows that for any $n \times p$ configuration Z, not just for solutions of the FDS stationary equations, if $V - B(Z)$ is indefinite we can decrease loss by adding another dimension. If Z is a stationary point and $V - B(Z)$ is positive semi-definite then we actually have found the CFDS solution, the Gower rank, and the global minimum (De Leeuw (2014)).

Penalizing Dimensions

In Shepard (1962a) and Shepard (1962b) a nonmetric multidimensional scaling technique is developed which minimizes a loss function over configurations in full dimensionality $n-1$. In that sense the technique is similar to FDS. Shepard's iterative process aims to maintain monotonicity between distances and dissimilarities and at the same time concentrate as much of the variation as possible in a small number of dimensions (De Leeuw (2017)).

Let us explore the idea of concentrating variation in $p < n-1$ dimensions, but use an approach which is quite different from the one used by Shepard. We remain in the FDS framework, but we aim for solutions in $p < n-1$ dimensions by penalizing $n-p$ dimensions of the full configuration, using the classical Courant quadratic penalty function.

Partition a full configuration $Z = \begin{bmatrix} X & \mid & Y \end{bmatrix}$, with X of dimension $n \times p$ and Y of dimension $n \times (n-p)$. Then

$$\sigma(Z) = 1 - \mathbf{tr}\ X'B(Z)X - \mathbf{tr}\ Y'B(Z)Y + \frac{1}{2}\mathbf{tr}\ X'VX + \frac{1}{2}\mathbf{tr}\ Y'VY. \tag{7}$$

Also define the *penalty term*

$$\tau(Y) = \frac{1}{2}\mathbf{tr}\ Y'VY, \tag{8}$$

and *penalized stress*

$$\pi(Z, \lambda) = \sigma(Z) + \lambda\ \tau(Y). \tag{9}$$

Our proposed method is to minimize penalized stress over Z for a sequence of values $0 = \lambda_1 < \lambda_2 < \cdots \lambda_m$. For $\lambda = 0$ this is simply the FDS problem, for which we know we can compute the global minimum. For fixed $0 < \lambda < +\infty$ this is a Penalized FDS or PFDS problem. PFDS problems with increasing values of λ generate a *trajectory* $Z(\lambda)$ in configuration space.

The general theory of exterior penalty functions, which we review in appendix A of this paper, shows that increasing λ leads to an increasing sequence of stress values σ and a decreasing sequence of penalty terms τ. If $\lambda \rightarrow +\infty$ we approximate the global minimum of the FDS problem with Z of the form $Z = \begin{bmatrix} X & \mid & 0 \end{bmatrix}$, i.e. of the pMDS problem. This assumes we do actually compute the global minimum for each value of λ, which we hope we can do because we start at the FDS global minimum, and we slowly increase λ. There is also a local version of the exterior penalty result, which implies that $\lambda \rightarrow \infty$ takes us to a local minimum of pMDS, so there is always the possibility of taking the wrong trajectory to a local minimum of pMDS.

Local Minima

The stationary equations of the PFDS problem are solutions to the equations

$$(V - B(Z))X = 0, \tag{10}$$

$$((1+\lambda)V - B(Z))Y = 0. \tag{11}$$

We can easily related stationary points and local minima of the FDS and PFDS problem.

Theorem 2: [PFDS Local Minima]

1: If X is a stationary point of the pMDS problem then $Z = [X \mid 0]$ is a stationary point of the PFDS problem, no matter what λ is.

2: If $Z = [X \mid 0]$ is a local minimum of the PFDS problem then X is a local minimum of pMDS and $(1+\lambda)V - B(X) \gtrsim 0$, or $\lambda \geq \|V^+B(X)\|_\infty - 1$, with $\| \bullet \|_\infty$ the spectral radius (largest eigenvalue).

Proof:

Part 1 follows by simple substitution in the stationary equations.

Part 2 follows from the expansion for $Z = [X + \epsilon P \mid \epsilon Q]$.

$$\pi(Z) = \pi(X) + \epsilon \text{ tr } P'\mathcal{D}\sigma(X) + \\ + \frac{1}{2}\epsilon^2\ \mathcal{D}^2\sigma(X)(P,P) + \frac{1}{2}\epsilon^2 \text{ tr } Q'((1+\lambda)V - B(X))Q + o(\epsilon^2). \tag{12}$$

At a local minimum we must have $\mathcal{D}\sigma(X) = 0$ and $\mathcal{D}^2\sigma(X)(P,P) \gtrsim 0$, which are the necessary conditions for a local minimum of pMDS. We also must have $((1+\lambda)V - B(X)) \gtrsim 0$. ■

Note that the conditions in part 2 of theorem 2 are also sufficient for PFDS to have a local minimum at $[X \mid 0]$, provided we eliminate translational and rotational indeterminacy by a suitable reparametrization, as in De Leeuw (1993).

Algorithm

The SMACOF algorithm for penalized stress is a small modification of the unpenalized FDS algorithm (5). We start our iterations for λ_j with the solution for λ_{j-1} (the starting solution for $\lambda_1 = 0$ can be completely arbitrary). The update rules for fixed λ are

$$X^{(k+1)} = V^+B(Z^{(k)})X^{(k)}, \tag{13}$$

$$Y^{(k+1)} = \frac{1}{1+\lambda}V^+B(Z^{(k)})Y^{(k)}. \tag{14}$$

Thus we compute the FDS update $Z^{(k+1)} = V^+B(Z^{(k)})Z^{(k)}$ and then divide the last $n-p$ columns by $1+\lambda$.

Code is in the appendix. Let us analyze a number of examples.

Examples

This section has a number of two-dimensional and a number of one-dimensional examples. The one-dimensional examples are of interest, because of the documented large number of local minima of stress in the one-dimensional case, and the fact that for small and medium n exact solutions are available (for example, De Leeuw (2005)). By default we use `seq(0, 1, length = 101)` for λ in most examples, but for some of them we dig a bit deeper and use longer sequences with smaller increments.

If for some value of λ the penalty term drops below the small cutoff γ, for example 10^{-10}, then there is not need to try larger values of λ, because they will just repeat the same result. We hope that result is the global minimum of the 2MDS problem.

The output for each example is a table in which we give, the minimum value of stress, the value of the penalty term at the minimum, the value of λ, and the number of iterations needed for convergence. Typically we print for the first three, the last three, and some regularly spaced intermediate values of λ. Remember that the stress values increase with increasing λ, and the penalty values decrease.

For two-dimensional examples we plot all two-dimensional configurations, after rotating to optimum match (using the function `matchMe()` from the appendix). We connect corresponding points for different values of λ. Points corresponding to the highest value of λ are labeled and have a different plot symbol. For one-dimensional examples we put `1:n` on the horizontal axes and plot the single dimension on the vertical axis, again connecting corresponding points. We label the points corresponding with the highest value of λ, and draw horizontal lines through them to more clearly show their order on the dimension.

The appendix also has code for the function `checkUni()`, which we have used to check the solutions in the one dimensional case are indeed local minima. The function checks the necessary condition for a local minimum $x = V^+u$, with

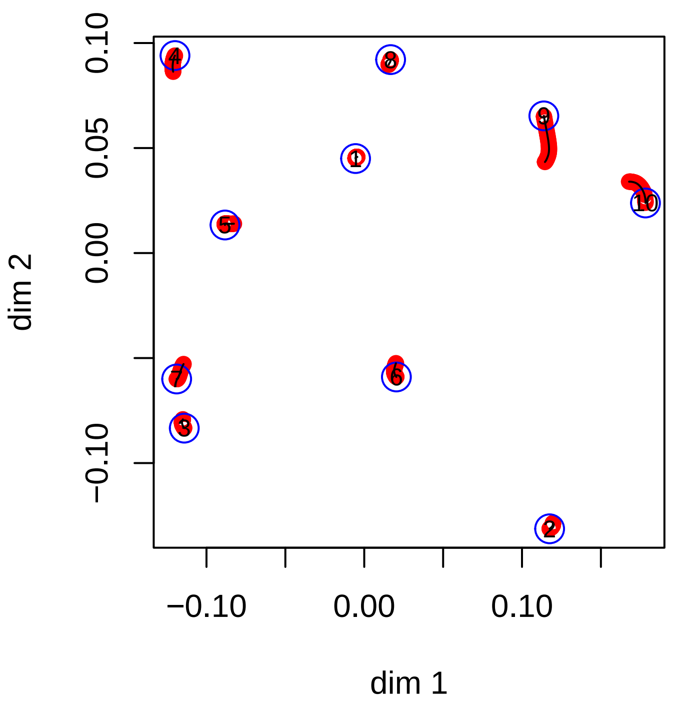
$$u_i = \sum_{j=1}^n w_{ij}\delta_{ij}\ \mathbf{sign}\ (x_i - x_j).$$

It should be emphasized that all examples are just meant to study convergence of penalized FDS. There is no interpretation of the MDS results

Chi Squares

In this example, of order 10, the δ_{ij} are independent draws from a chi-square distribution with two degrees of freedom. There is no structure in this example, everything is random.

```
## itel  198 lambda   0.000000 stress 0.175144 penalty 0.321138
## itel    5 lambda   0.010000 stress 0.175156 penalty 0.027580
## itel    3 lambda   0.020000 stress 0.175187 penalty 0.025895
## itel    1 lambda   0.100000 stress 0.175914 penalty 0.015172
## itel    1 lambda   0.200000 stress 0.177666 penalty 0.004941
## itel    4 lambda   0.300000 stress 0.178912 penalty 0.000088
## itel    6 lambda   0.310000 stress 0.178933 penalty 0.000020
## itel   20 lambda   0.320000 stress 0.178939 penalty 0.000000
```

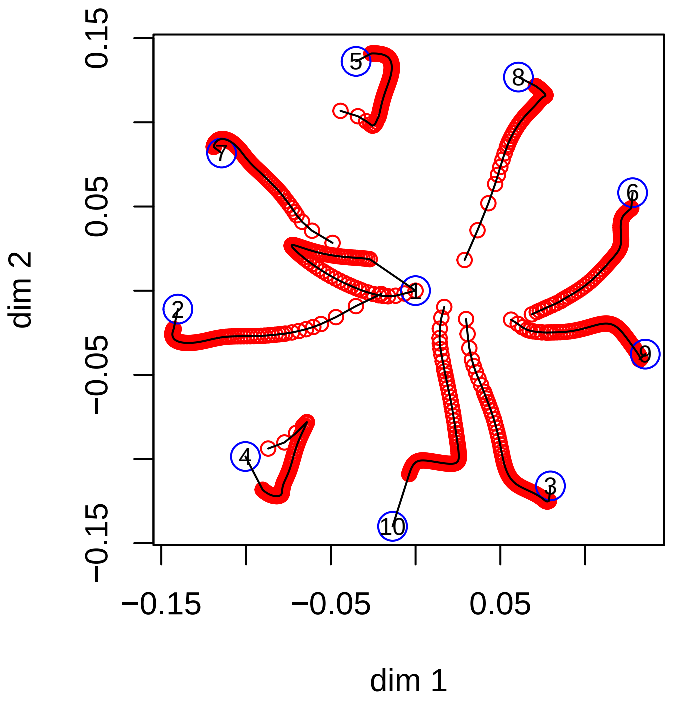


It seems that in this example the first two dimensions of FDS are already close to optimal for 2MDS. This is because the Gower rank of the dissimilarities is only three (or maybe four, the fourth singular value of the FDS solution Z is very small).

Regular Simplex

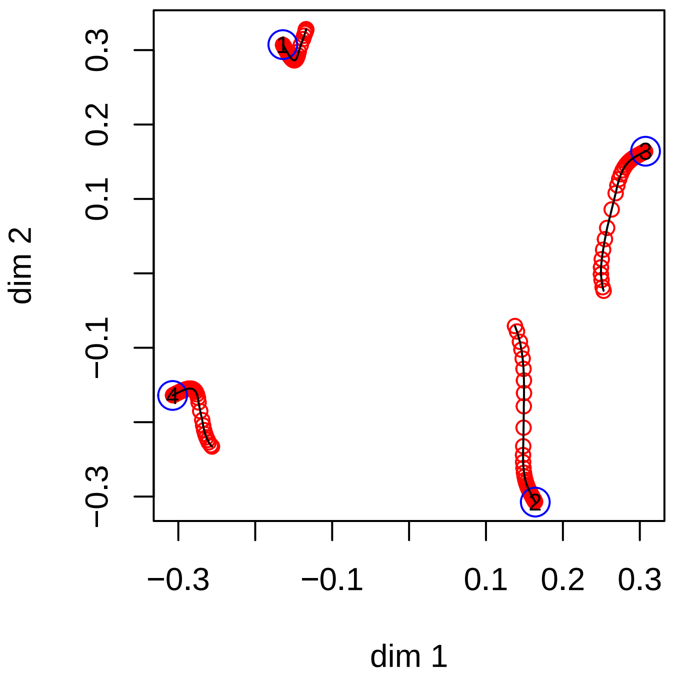
The regular simplex has all dissimilarities equal to one. We use an example with $n = 10$, for which the global minimum (as far as we know) of pMDS with $p = 2$ is a configuration with nine points equally spaced on a circle and one point in the center.

```
## itel    1 lambda   0.000000 stress 0.000000 penalty 0.400000
## itel    7 lambda   0.010000 stress 0.000099 penalty 0.376711
## itel    5 lambda   0.020000 stress 0.000406 penalty 0.362565
## itel    1 lambda   0.100000 stress 0.009083 penalty 0.257421
## itel    1 lambda   0.200000 stress 0.032156 penalty 0.149837
## itel    1 lambda   0.300000 stress 0.060372 penalty 0.075680
## itel    1 lambda   0.400000 stress 0.082108 penalty 0.037070
## itel    1 lambda   0.500000 stress 0.097151 penalty 0.016156
## itel    1 lambda   0.600000 stress 0.106607 penalty 0.004838
## itel    1 lambda   0.690000 stress 0.109990 penalty 0.000964
## itel    1 lambda   0.700000 stress 0.110133 penalty 0.000771
## itel  106 lambda   0.710000 stress 0.109880 penalty 0.000000
```



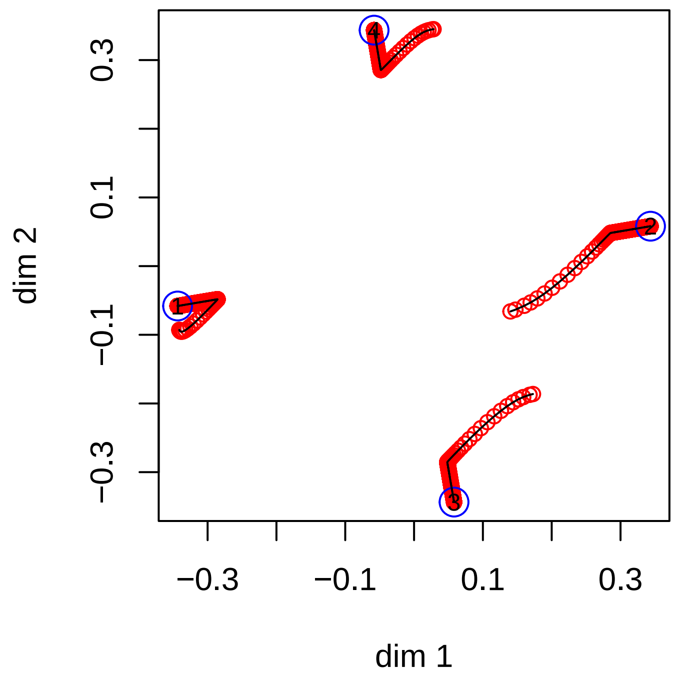
Next, we look at the regular simplex with $n = 4$, for which the global minimum has four points equally spaced on a circle (i.e. in the corners of a square). We use `seq(0, 1, length = 101)` for the λ sequence.

```
## itel    1 lambda   0.000000 stress 0.000000 penalty 0.250000
## itel    2 lambda   0.010000 stress 0.000033 penalty 0.162166
## itel    2 lambda   0.020000 stress 0.000158 penalty 0.156683
## itel    2 lambda   0.100000 stress 0.004569 penalty 0.103311
## itel    1 lambda   0.200000 stress 0.016120 penalty 0.041571
## itel    1 lambda   0.300000 stress 0.026039 penalty 0.007635
## itel    1 lambda   0.330000 stress 0.027355 penalty 0.003658
## itel    5 lambda   0.340000 stress 0.028272 penalty 0.000944
## itel   36 lambda   0.350000 stress 0.028595 penalty 0.000000
```



The solution converges to an equilateral triangle with the fourth point in the centroid. This is a local minimum. What basically happens is that the first two dimensions of the FDS solution are too close to the local minimum. Or, what amounts to the same thing, the Gower rank is too large (it is $n-1$ for a regular simplex) , there is too much variation in the higher dimensions, and as a consequence the first two dimensions of FDS are a bad 2MDS solution. We try to repair this by refining the trajectory, using `seq(0, 1, 10001)`.

```
## itel    1 lambda   0.000000 stress 0.000000 penalty 0.250000
## itel    2 lambda   0.000100 stress 0.000000 penalty 0.166621
## itel    2 lambda   0.000200 stress 0.000000 penalty 0.166563
## itel    1 lambda   0.202200 stress 0.028595 penalty 0.000001
## itel    1 lambda   0.202300 stress 0.028595 penalty 0.000001
## itel    1 lambda   0.202400 stress 0.028595 penalty 0.000001
```

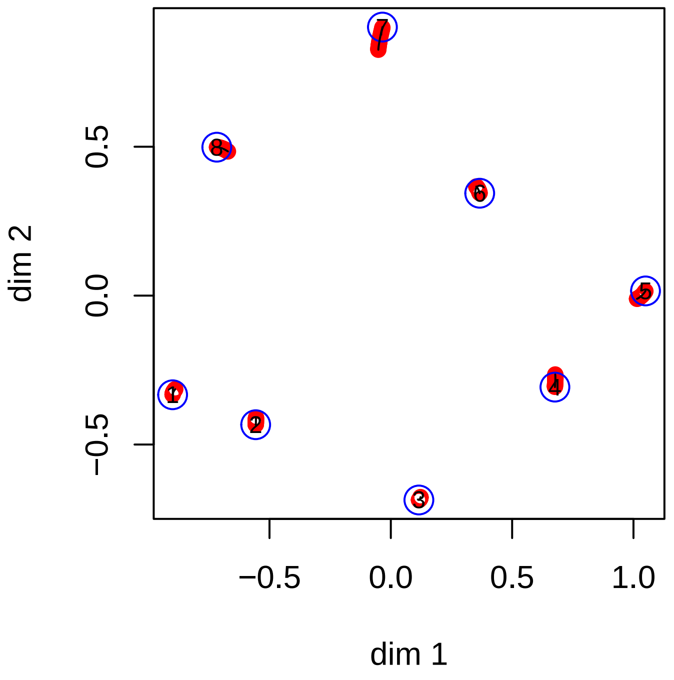


Now the trajectories move us from what starts out similar to an equilateral triangle to the corners of the square, and thus we do find the global minimum in this way. It is remarkable that we manage to find the square even when we start closer to the triangle with midpoint.

Intelligence

These are correlations between eight intelligence tests, taken from the `smacof` package. We convert to dissimilarities by taking the negative logarithm of the correlations. As in the chi-square example, the FDS and the 2MDS solution are very similar and the PMDS trajectories are short.

```
## itel 2951 lambda   0.000000 stress 0.107184 penalty 7.988384
## itel    7 lambda   0.010000 stress 0.107560 penalty 0.685654
## itel    4 lambda   0.020000 stress 0.108528 penalty 0.628538
## itel    3 lambda   0.030000 stress 0.110045 penalty 0.573208
## itel    3 lambda   0.040000 stress 0.112449 penalty 0.510730
## itel    2 lambda   0.050000 stress 0.114714 penalty 0.464650
## itel    2 lambda   0.060000 stress 0.117623 penalty 0.415037
## itel    2 lambda   0.070000 stress 0.121095 penalty 0.364536
## itel    2 lambda   0.080000 stress 0.125010 penalty 0.315023
## itel    2 lambda   0.090000 stress 0.129226 penalty 0.267831
## itel    2 lambda   0.100000 stress 0.133589 penalty 0.223898
## itel    2 lambda   0.110000 stress 0.137944 penalty 0.183868
## itel    3 lambda   0.120000 stress 0.143921 penalty 0.133739
## itel    2 lambda   0.130000 stress 0.147473 penalty 0.106166
## itel    4 lambda   0.140000 stress 0.153215 penalty 0.064499
## itel    4 lambda   0.150000 stress 0.157159 penalty 0.037735
## itel    9 lambda   0.160000 stress 0.161434 penalty 0.010337
## itel   72 lambda   0.170000 stress 0.163122 penalty 0.000000
```

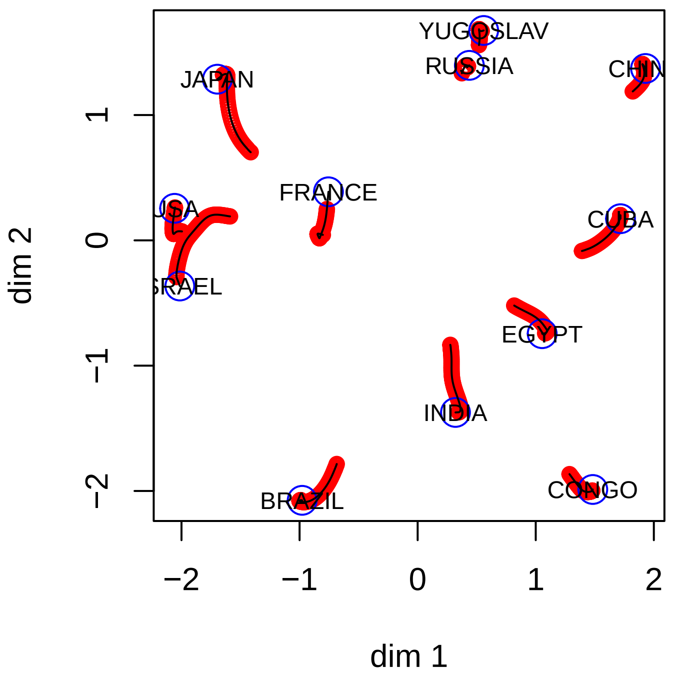


The singular values of the FDS solution are 1.78e+00, 1.36e+00, 5.94e-01, 1.47e-01, 3.29e-03, 1.06e-16, 3.55e-17, 3.09e-17, which shows that the Gower rank is probably five, but approximately two.

Countries

This is the `wish` dataset from the 'smacof' package, with similarities between 12 countries. They are converted to dissimilarties by subtracting each of them from seven.

```
## itel 1381 lambda   0.000000 stress 4.290534 penalty 98.617909
## itel    4 lambda   0.010000 stress 4.301341 penalty 36.137074
## itel    3 lambda   0.020000 stress 4.336243 penalty 34.389851
## itel    1 lambda   0.100000 stress 5.187917 penalty 23.300775
## itel    1 lambda   0.200000 stress 7.539228 penalty 11.543635
## itel    1 lambda   0.300000 stress 9.901995 penalty 4.963372
## itel    1 lambda   0.400000 stress 11.523357 penalty 1.859569
## itel    1 lambda   0.500000 stress 12.391692 penalty 0.556411
## itel    1 lambda   0.590000 stress 12.696493 penalty 0.080144
## itel    1 lambda   0.600000 stress 12.708627 penalty 0.060113
## itel  100 lambda   0.610000 stress 12.738355 penalty 0.000000
```



The singular values of the FDS solution are 4.20e+00, 3.71e+00, 2.67e+00, 1.80e+00, 1.33e+00, 6.64e-01, 5.97e-04, 5.59e-16, 3.59e-16, 2.25e-16, 1.80e-16, 2.47e-17, and the Gower rank is six or seven.

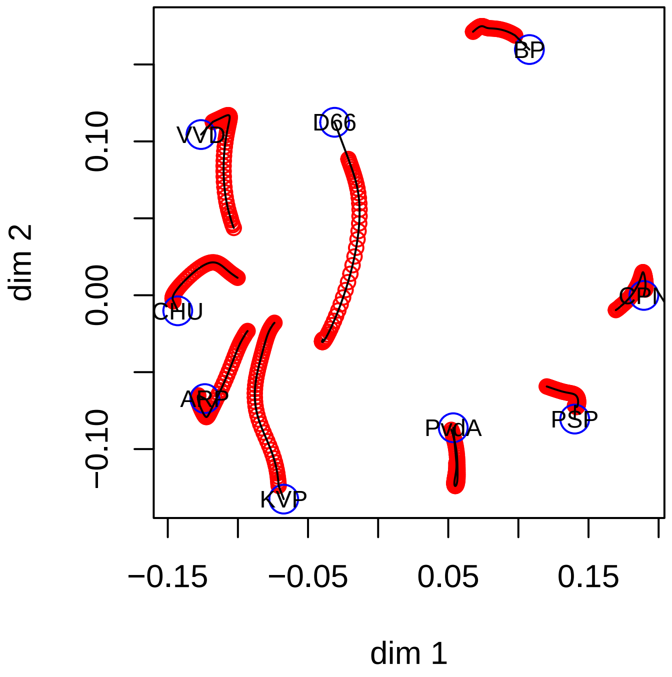
Dutch Political Parties

In 1967 one hundred psychology students at Leiden University judged the similarity of nine Dutch political parties, using the complete method of triads (De Gruijter (1967)). Data were aggregated and converted to dissimilarities. We first print the matrix of dissimilarities.

```
##       KVP   PvdA  VVD   ARP   CHU   CPN   PSP   BP    D66
## KVP   0.000 0.209 0.196 0.171 0.179 0.281 0.250 0.267 0.230
## PvdA  0.209 0.000 0.250 0.210 0.231 0.190 0.171 0.269 0.204
## VVD   0.196 0.250 0.000 0.203 0.185 0.302 0.281 0.257 0.174
## ARP   0.171 0.210 0.203 0.000 0.119 0.292 0.250 0.271 0.228
## CHU   0.179 0.231 0.185 0.119 0.000 0.290 0.263 0.259 0.225
## CPN   0.281 0.190 0.302 0.292 0.290 0.000 0.152 0.236 0.276
## PSP   0.250 0.171 0.281 0.250 0.263 0.152 0.000 0.256 0.237
## BP    0.267 0.269 0.257 0.271 0.259 0.236 0.256 0.000 0.274
## D66   0.230 0.204 0.174 0.228 0.225 0.276 0.237 0.274 0.000
```

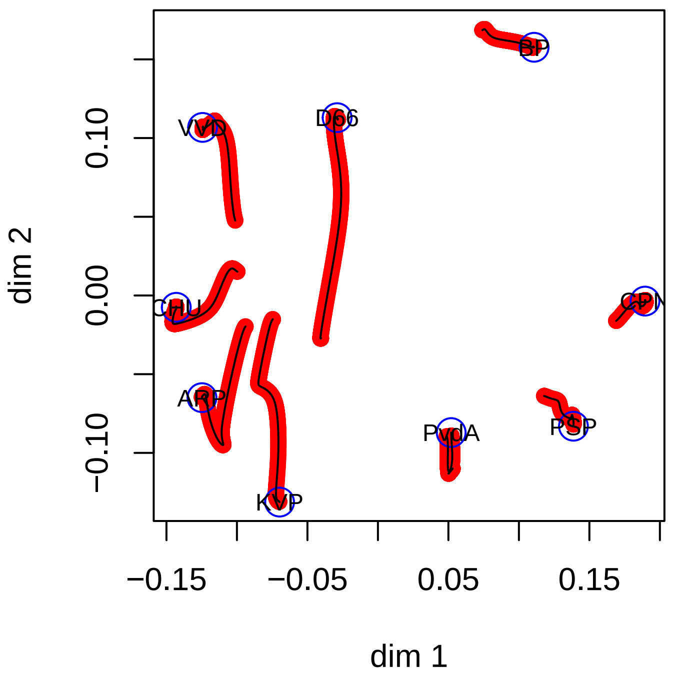
The trajectories from FDS to 2MDS show some clear movement, especially of the D’66 party, which was new at the time.

```
## itel  223 lambda   0.000000 stress 0.000000 penalty 0.414526
## itel    5 lambda   0.010000 stress 0.000061 penalty 0.196788
## itel    2 lambda   0.020000 stress 0.000199 penalty 0.190472
## itel    1 lambda   0.100000 stress 0.004399 penalty 0.136576
## itel    1 lambda   0.200000 stress 0.015811 penalty 0.075466
## itel    1 lambda   0.300000 stress 0.028235 penalty 0.036636
## itel    1 lambda   0.400000 stress 0.038275 penalty 0.012608
## itel    1 lambda   0.500000 stress 0.043644 penalty 0.002156
## itel    1 lambda   0.520000 stress 0.044091 penalty 0.001324
## itel    1 lambda   0.530000 stress 0.044253 penalty 0.001019
## itel  277 lambda   0.540000 stress 0.044603 penalty 0.000000
```



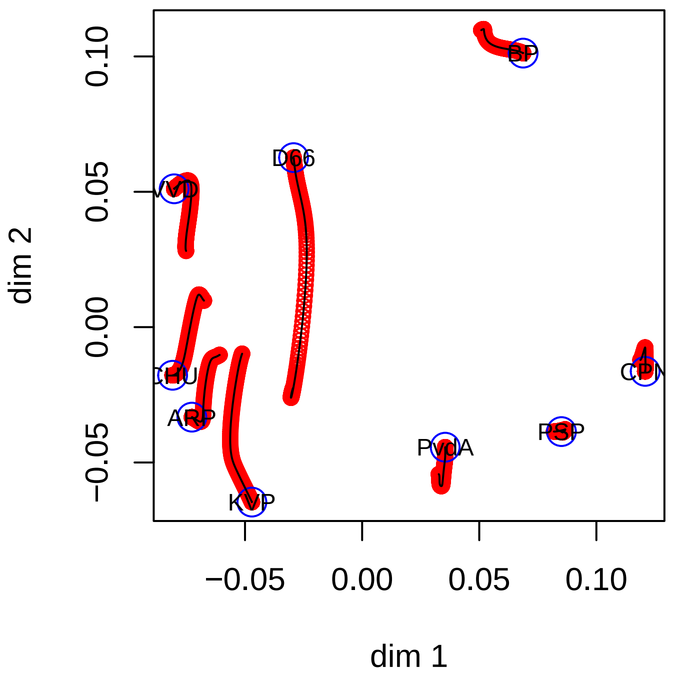
There seems to be some bifurcation going on at the end, so we repeat the analysis using `seq(0, 1, length = 1001)` for λ. The results turn out to be basically the same.

```
## itel  223 lambda   0.000000 stress 0.000000 penalty 0.414526
## itel    4 lambda   0.001000 stress 0.000001 penalty 0.204225
## itel    2 lambda   0.002000 stress 0.000002 penalty 0.203535
## itel    1 lambda   0.468000 stress 0.044604 penalty 0.000001
## itel    1 lambda   0.469000 stress 0.044604 penalty 0.000000
## itel  166 lambda   0.470000 stress 0.044603 penalty 0.000000
```



The singular values of the FDS solution are 2.95e-01, 2.10e-01, 1.89e-01, 1.34e-01, 1.16e-01, 1.06e-01, 8.61e-02, 7.06e-02, 7.77e-19, and the Gower rank is probably eight. This is mainly because these data, being averages, regress to the mean and thus have a substantial additive constant. If we repeat the analysis after subtracting .1 from all dissimilarities we get basically the same solution, but with somewhat smoother trajectories.

```
## itel  511 lambda   0.000000 stress 0.000176 penalty 0.150789
## itel    2 lambda   0.001000 stress 0.000176 penalty 0.037759
## itel    2 lambda   0.002000 stress 0.000176 penalty 0.037619
## itel    1 lambda   0.370000 stress 0.007642 penalty 0.000000
## itel    1 lambda   0.371000 stress 0.007642 penalty 0.000000
## itel    1 lambda   0.372000 stress 0.007642 penalty 0.000000
```

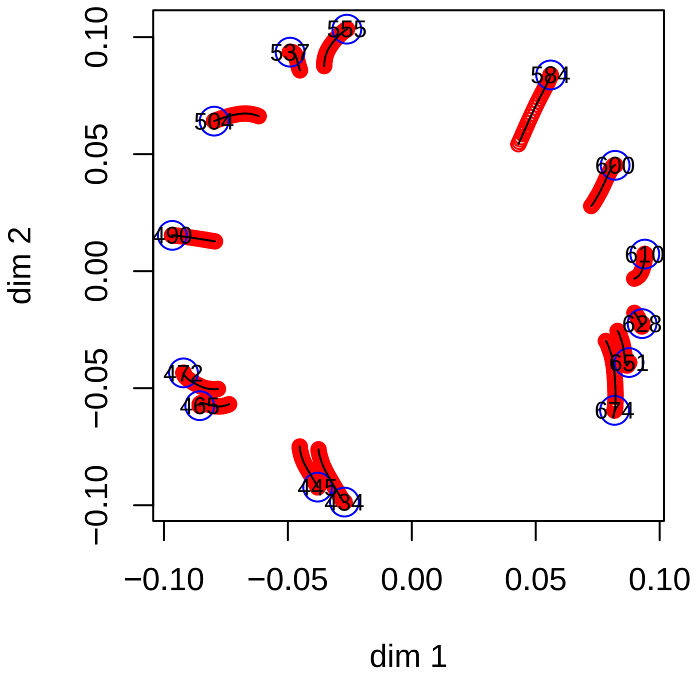


Now the singular values of the FDS solution are 2.05e-01, 1.34e-01, 1.11e-01, 6.03e-02, 3.11e-02, 3.97e-04, 1.18e-07, 4.55e-12, 1.06e-17, and the approximate Gower rank is more like five or six.

Ekman

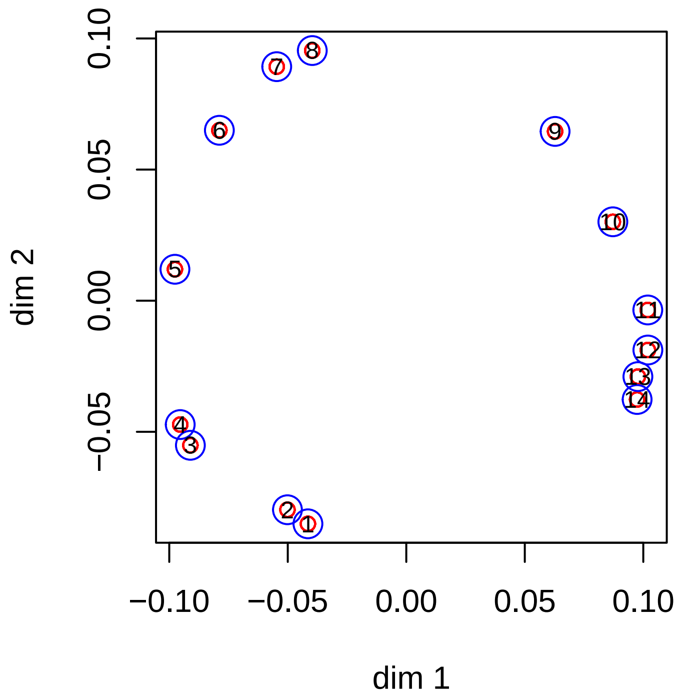
The next example analyzes dissimilarities between 14 colors, taken from Ekman (1954). The original similarities s_{ij}, averaged over 31 subjects, were transformed to dissimilarities by $\delta_{ij} = 1 - s_{ij}$.

```
## itel 1482 lambda   0.000000 stress 0.000088 penalty 0.426110
## itel    5 lambda   0.010000 stress 0.000132 penalty 0.118988
## itel    3 lambda   0.020000 stress 0.000253 penalty 0.112777
## itel    1 lambda   0.100000 stress 0.003195 penalty 0.070791
## itel    1 lambda   0.200000 stress 0.010778 penalty 0.024407
## itel    1 lambda   0.300000 stress 0.016125 penalty 0.003230
## itel    1 lambda   0.400000 stress 0.017142 penalty 0.000165
## itel    4 lambda   0.500000 stress 0.017213 penalty 0.000000
## itel    1 lambda   0.600000 stress 0.017213 penalty 0.000000
## itel    1 lambda   0.610000 stress 0.017213 penalty 0.000000
## itel    1 lambda   0.620000 stress 0.017213 penalty 0.000000
## itel    1 lambda   0.630000 stress 0.017213 penalty 0.000000
```



If we transform the Ekman similarities by $\delta_{ij} = (1 - s_{ij})^3$ then its is known (De Leeuw (2016)) that the Gower rank is equal to two. Thus the FDS solution has rank 2, and the 2MDS solution is the global minimum.

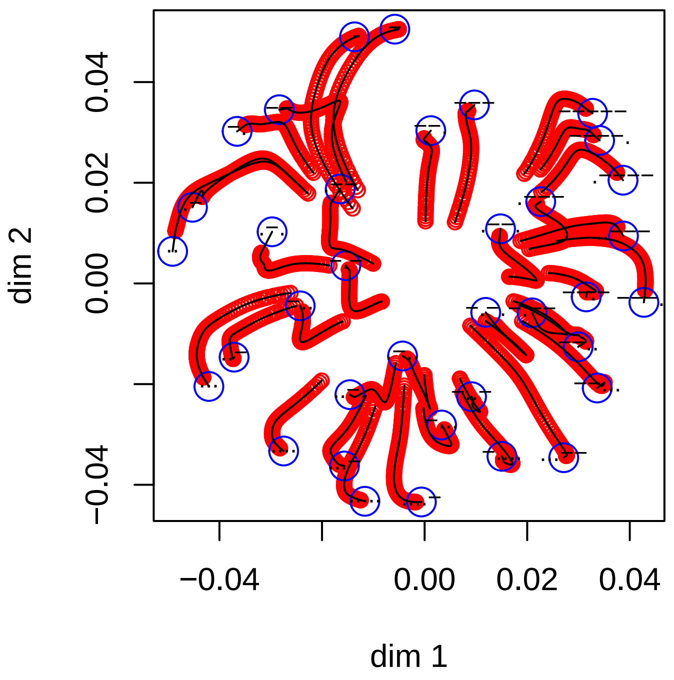
```
## itel   99 lambda   0.000000 stress 0.011025 penalty 0.433456
## itel    1 lambda   0.010000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.020000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.100000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.110000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.120000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.130000 stress 0.011025 penalty 0.000000
```



Morse in Two

Next, we use dissimilarities between 36 Morse code signals (Rothkopf (1957)). We used the symmetrized version `morse` from the `smacof` package (De Leeuw and Mair (2009)).

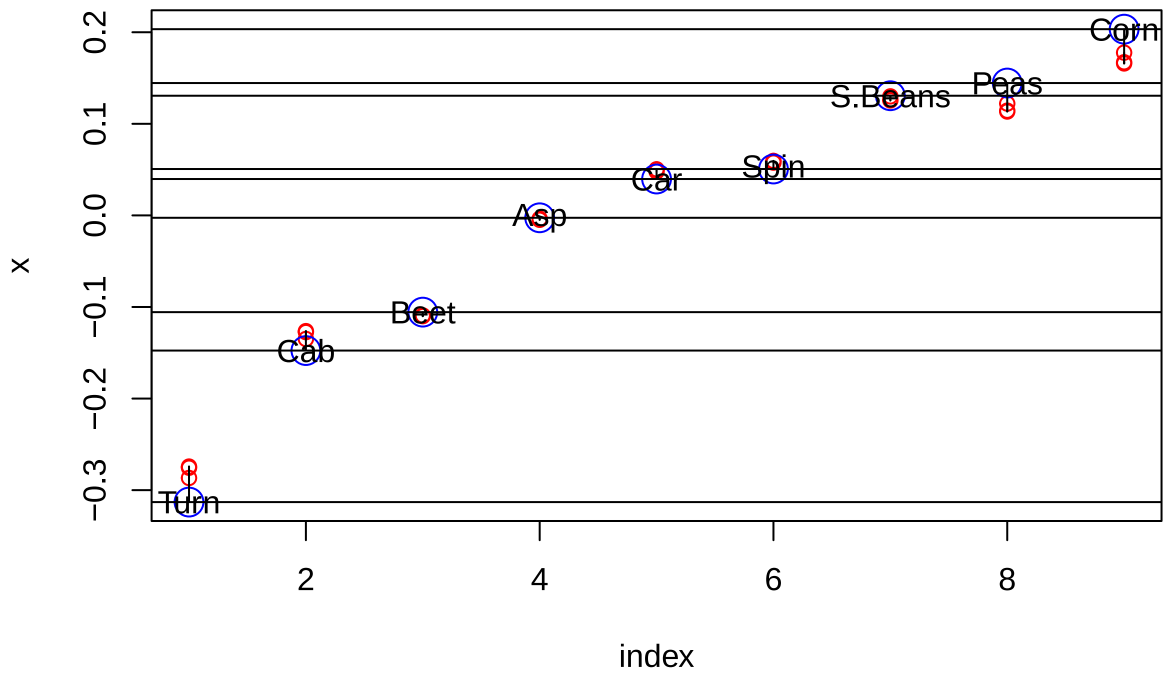
```
## itel 1461 lambda   0.000000 stress 0.000763 penalty 0.472254
## itel    6 lambda   0.010000 stress 0.000858 penalty 0.322181
## itel    4 lambda   0.020000 stress 0.001147 penalty 0.308335
## itel    1 lambda   0.100000 stress 0.008576 penalty 0.216089
## itel    1 lambda   0.200000 stress 0.028903 penalty 0.119364
## itel    1 lambda   0.300000 stress 0.051285 penalty 0.060060
## itel    1 lambda   0.400000 stress 0.068653 penalty 0.028190
## itel    1 lambda   0.500000 stress 0.080258 penalty 0.011356
## itel    1 lambda   0.600000 stress 0.086572 penalty 0.003578
## itel    1 lambda   0.700000 stress 0.089140 penalty 0.000854
## itel    1 lambda   0.800000 stress 0.089898 penalty 0.000116
## itel    1 lambda   0.830000 stress 0.089958 penalty 0.000053
## itel    1 lambda   0.840000 stress 0.089970 penalty 0.000040
## itel  197 lambda   0.850000 stress 0.089949 penalty 0.000000
```



Vegetables

Our first one-dimensional example uses paired comparisons of 9 vegetables, originating with Guilford (1954) and taken from the `psych` package (Revelle (2018)). The proportions are transformed to dissimilarities by using the normal quantile function, i.e. $\delta_{ij} = |\Phi^{-1}(p_{ij})|$. We use a short sequence for λ.

```
## itel 1412 lambda   0.000000 stress 0.013675 penalty 0.269308
## itel    5 lambda   0.010000 stress 0.013716 penalty 0.114786
## itel    5 lambda   0.100000 stress 0.016719 penalty 0.069309
## itel   23 lambda   1.000000 stress 0.035301 penalty 0.000000
```

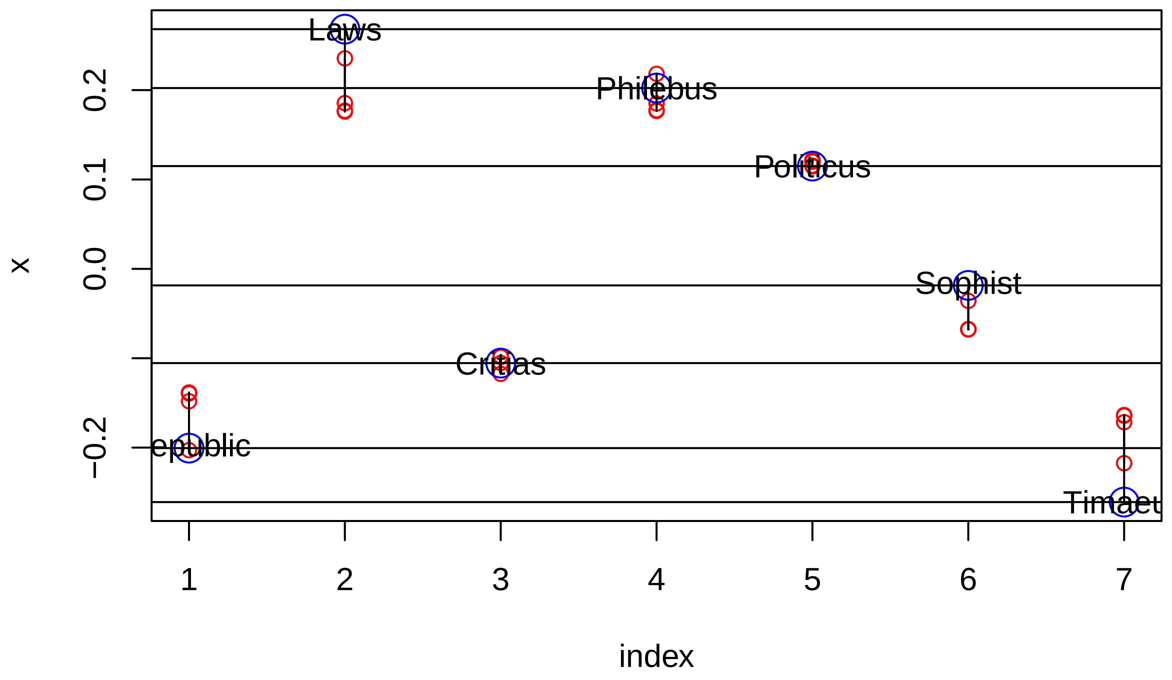


This example was previously analyzed in De Leeuw (2005) using enumeration of all permutations. He found 14354 isolated local minima, and a global minimum equal to the one we computed here.

Plato

Mair, Groenen, and De Leeuw (2022) use seriation of the works of Plato, from the data collected by Cox and Brandwood (1959), as an example of unidimensional scaling. We first run this example with our usual sequence of five λ values.

```
## itel  169 lambda   0.000000 stress 0.000000 penalty 0.410927
## itel    3 lambda   0.010000 stress 0.000062 penalty 0.255246
## itel    3 lambda   0.100000 stress 0.005117 penalty 0.194993
## itel    4 lambda   1.000000 stress 0.106058 penalty 0.019675
## itel    9 lambda  10.000000 stress 0.139462 penalty 0.000000
```

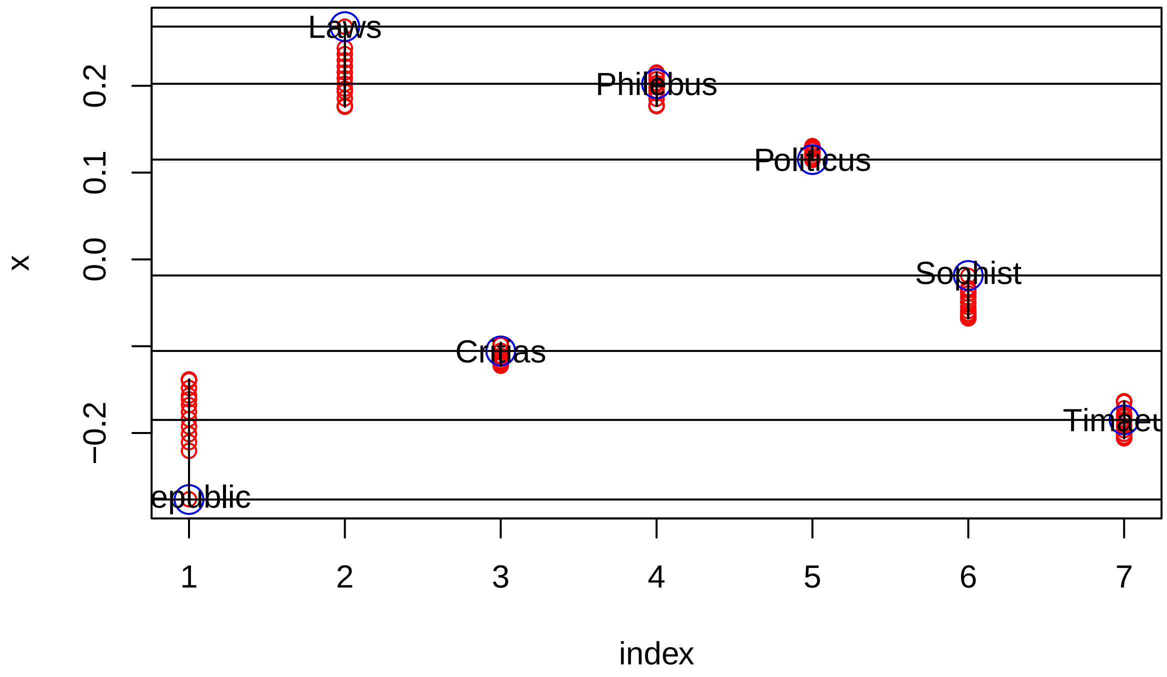


This gives the order

```
##      [,1]
## [1,] "Timaeus"
## [2,] "Republic"
## [3,] "Critias"
## [4,] "Sophist"
## [5,] "Politicus"
## [6,] "Philebus"
## [7,] "Laws"
```

which is different from the order at the global minimum, which has Republic before Timaeus. Thus we have recovered a local minimum, and it seems our sequence of λ values was not fine enough to do the job properly. So we try a longer and finer sequence.

```
## itel  169 lambda   0.000000 stress 0.000000 penalty 0.410927
## itel    3 lambda   0.000100 stress 0.000000 penalty 0.263015
## itel    3 lambda   0.001000 stress 0.000001 penalty 0.262280
## itel    3 lambda   0.010000 stress 0.000064 penalty 0.255078
## itel    3 lambda   0.100000 stress 0.005123 penalty 0.194945
## itel    2 lambda   0.200000 stress 0.016184 penalty 0.147493
## itel    1 lambda   0.300000 stress 0.026997 penalty 0.119323
## itel    1 lambda   0.400000 stress 0.040023 penalty 0.093615
## itel    1 lambda   0.500000 stress 0.053688 penalty 0.072330
## itel    1 lambda   0.600000 stress 0.066833 penalty 0.055452
## itel    1 lambda   0.700000 stress 0.078832 penalty 0.042269
## itel    1 lambda   0.800000 stress 0.089439 penalty 0.032019
## itel    1 lambda   0.900000 stress 0.098557 penalty 0.024079
## itel    1 lambda   1.000000 stress 0.106135 penalty 0.017940
## itel    6 lambda   2.000000 stress 0.130789 penalty 0.000148
## itel   13 lambda   3.000000 stress 0.131135 penalty 0.000000
```



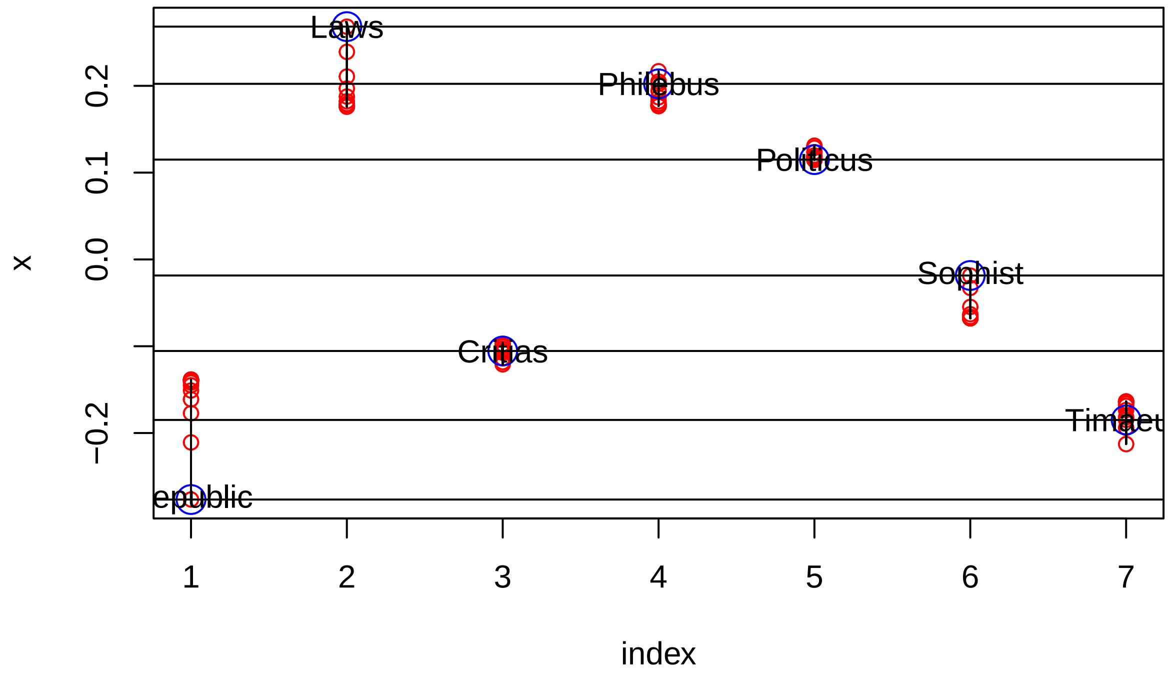
Now the order is

```
##      [,1]
## [1,] "Republic"
## [2,] "Timaeus"
## [3,] "Critias"
## [4,] "Sophist"
## [5,] "Politicus"
## [6,] "Philebus"
## [7,] "Laws"
```

which does indeed correspond to the global minimum.

With a different λ sequence we find the same solution.

```
## itel  169 lambda   0.000000 stress 0.000000 penalty 0.410927
## itel    3 lambda   0.001000 stress 0.000001 penalty 0.262296
## itel    2 lambda   0.002000 stress 0.000003 penalty 0.261483
## itel    2 lambda   0.004000 stress 0.000010 penalty 0.259872
## itel    2 lambda   0.008000 stress 0.000041 penalty 0.256690
## itel    2 lambda   0.016000 stress 0.000159 penalty 0.250470
## itel    2 lambda   0.032000 stress 0.000613 penalty 0.238574
## itel    2 lambda   0.064000 stress 0.002266 penalty 0.216785
## itel    2 lambda   0.128000 stress 0.007791 penalty 0.180067
## itel    2 lambda   0.256000 stress 0.023483 penalty 0.127006
## itel    2 lambda   0.512000 stress 0.056940 penalty 0.067948
## itel    3 lambda   1.024000 stress 0.107743 penalty 0.017937
## itel    8 lambda   2.048000 stress 0.131059 penalty 0.000032
## itel    9 lambda   4.096000 stress 0.131135 penalty 0.000000
```



The order is

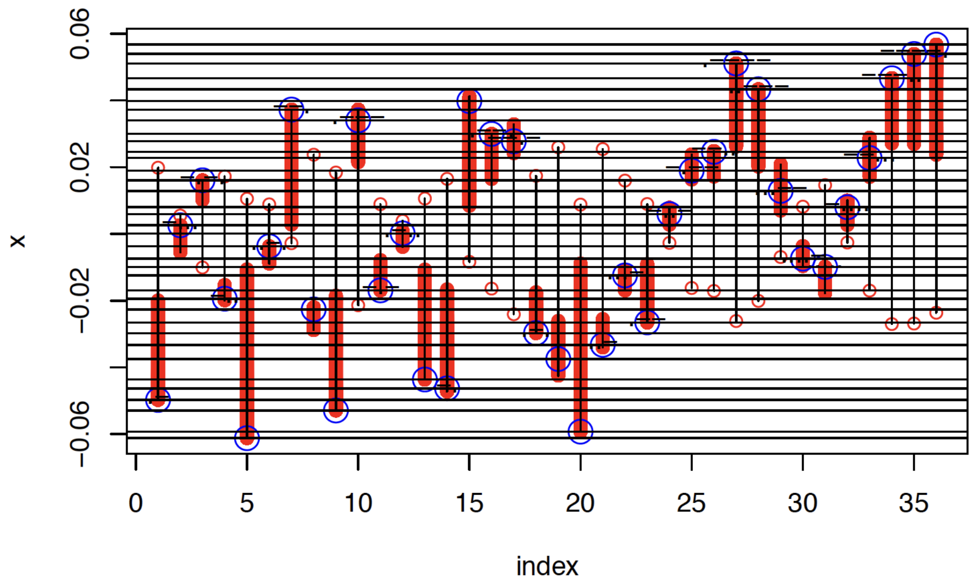
```
##      [,1]
## [1,] "Republic"
## [2,] "Timaeus"
## [3,] "Critias"
## [4,] "Sophist"
## [5,] "Politicus"
## [6,] "Philebus"
## [7,] "Laws"
```

Morse in One

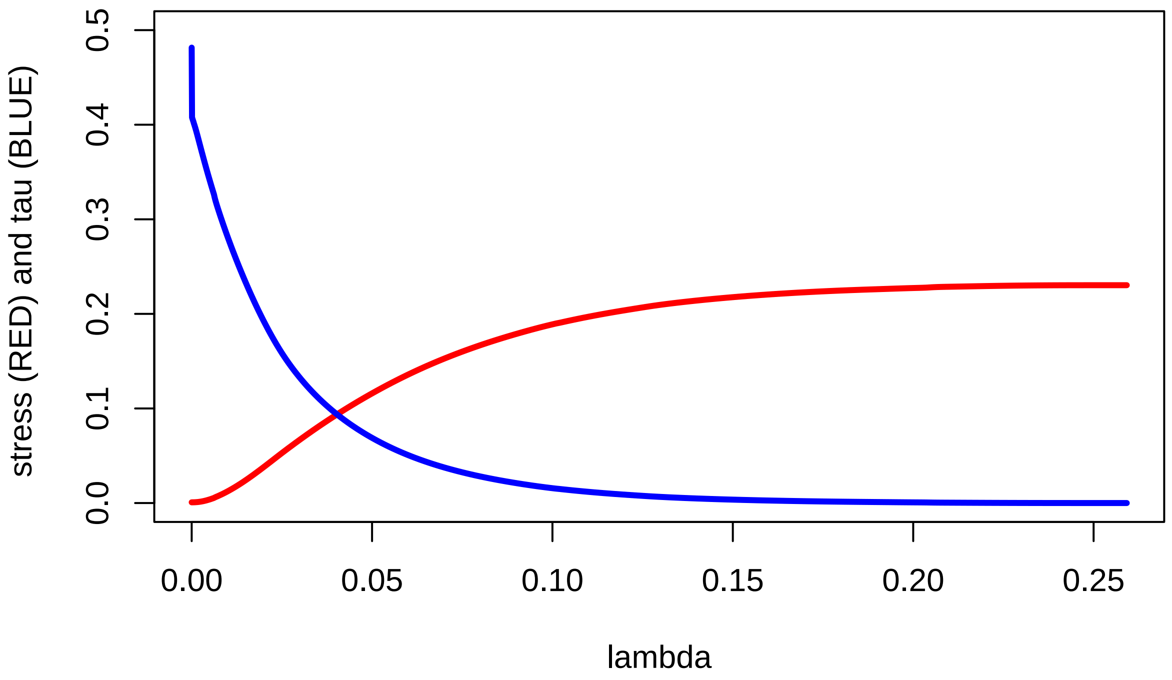
Now for a more challenging example. The Morse code data have been used to try out exact unidimensional MDS techniques, for example by Palubeckis (2013). We will enter the global minimum contest by using 10,000 values of λ, in an equally spaced sequence from 0 to 10. This is not as bad as it sounds. For the 10,000 FDS solutions `system.time()` tells us

```
##    user  system elapsed
##   4.872   0.277   5.152
```

The one-dimensional plot show quite a bit of movement, but much of it seems to be contained in the very first change of λ.



We can also plot stress and the penalty term as functions of λ. Again, note the big change in the penalty term when λ goes from zero to 0.001.



After the first 2593 values of λ the penalty term is zero and we stop, i.e. we estimate λ_+ is 2.593. At that point we have run a total of 5013 FDS iterations, and thus on average about two iterations per λ value. Stress has increased from 0.0007634501 to 0.2303106976 and the penalty value has decreased from 0.4815136419 to 0.0000000001. We find the following order of the points on the dimension.

```
##       [,1]
##  [1,] "."
##  [2,] "-"
##  [3,] ".."
##  [4,] ".-"
##  [5,] "-."
##  [6,] "--"
##  [7,] "..."
##  [8,] "..-"
##  [9,] ".-."
## [10,] ".--"
## [11,] "...."
## [12,] "-.."
## [13,] "-.-"
## [14,] "...-"
## [15,] "....."
## [16,] "....-"
## [17,] "..-."
## [18,] ".-.."
## [19,] "-..."
## [20,] "-..-"
## [21,] "-...."
## [22,] "...--"
## [23,] "-.-."
## [24,] "-.--"
## [25,] "--..."
## [26,] "--.."
## [27,] "--.-"
## [28,] ".--."
## [29,] ".---"
## [30,] "--."
## [31,] "---"
## [32,] "..---"
## [33,] "---.."
## [34,] ".----"
## [35,] "----."
## [36,] "-----"
```

Our order, and consequently our solution, is the same as the exact global solution given by Palubeckis (2013). See his table 4, reproduced below. The difference is that computing our solution takes 10 seconds, while his takes 494 seconds. But off course we would not know we actually found the global mimimum if the exact exhaustive methods had not analyzed the same data before.

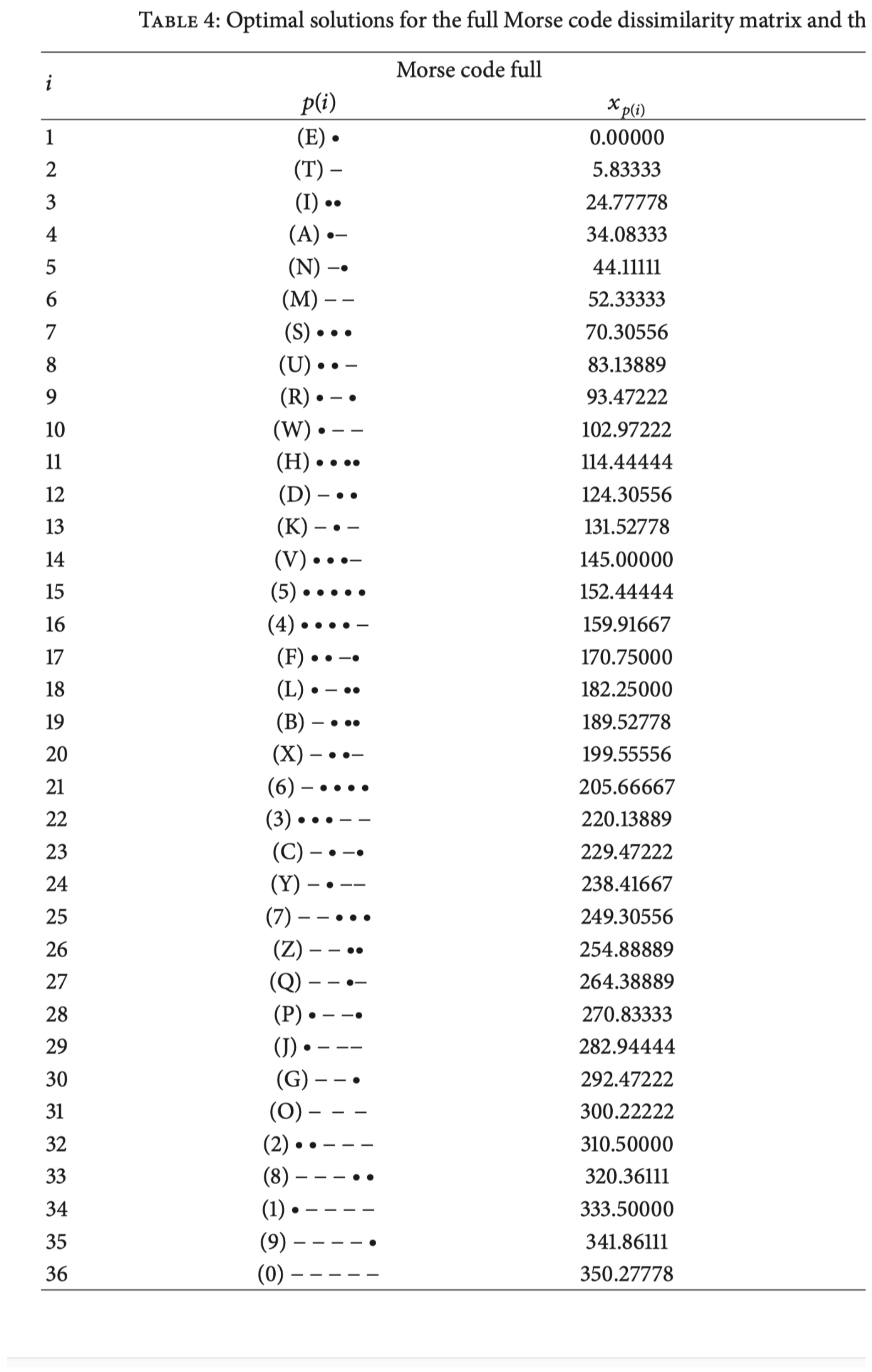


TABLE 4: Optimal solutions for the full Morse code dissimilarity matrix and th

i	Morse code full	
	$p(i)$	$x_{p(i)}$
1	(E) •	0.00000
2	(T) –	5.83333
3	(I) ••	24.77778
4	(A) •–	34.08333
5	(N) –•	44.11111
6	(M) – –	52.33333
7	(S) • • •	70.30556
8	(U) • • –	83.13889
9	(R) • – •	93.47222
10	(W) • – –	102.97222
11	(H) • • ••	114.44444
12	(D) – • •	124.30556
13	(K) – • –	131.52778
14	(V) • • •–	145.00000
15	(5) • • • • •	152.44444
16	(4) • • • • –	159.91667
17	(F) • • –•	170.75000
18	(L) • – ••	182.25000
19	(B) – • ••	189.52778
20	(X) – • •–	199.55556
21	(6) – • • • •	205.66667
22	(3) • • • – –	220.13889
23	(C) – • –•	229.47222
24	(Y) – • ––	238.41667
25	(7) – – • • •	249.30556
26	(Z) – – ••	254.88889
27	(Q) – – •–	264.38889
28	(P) • – –•	270.83333
29	(J) • – ––	282.94444
30	(G) – – •	292.47222
31	(O) – – –	300.22222
32	(2) • • – – –	310.50000
33	(8) – – – • •	320.36111
34	(1) • – – – –	333.50000
35	(9) – – – – •	341.86111
36	(0) – – – – –	350.27778

Discussion

There is one surprising (to me, at least) finding from all our examples. There is a value, say λ_+, such that the penalty $\tau(Y)$ is zero for all PFDS solutions with $\lambda \geq \lambda_+$. In other words, our penalty function acts like a *smooth exact penalty function.* The precise reason for exactness in our case (if there is one) is not entirely clear to me yet, but it is obviously a topic for further research, using for example the recent theoretical framework of Dolgopolik (2016a), Dolgopolik (2016b), Dolgopolik (2017), Dolgopolik (n.d.).

In our two dimensional examples we always start our plots with the first two dimensions of the FDS configuration. These two-dimensional configurations are usually small (all points relatively close to the origin), because so much variation is still in the higher dimensions. If λ increases the growth of the configurations is one important aspect of configuration change.

In our iteration counts with short sequences of λ we see relatively small increases in stress and small decreases in the penalty term, until we get closer to λ_+, when we suddenly see a sudden change and a larger number of iterations. This is also reflected in the figures, where generally the change to the last solution (with the largest λ) makes the largest jump. This suggest a finer sequence near λ_+ and perhaps an adaptive strategy for choosing λ. Or to use brute force, as in the unidimensional Morse code example. With such longer and finer sequences convergence becomes more smooth.

Another all-important aspect of the method discussed here is that it assumes computation of the global minimum for each λ. Since we cannot expect a result as nice as the one for FDS (all local minima are global) for $\lambda > 0$ our method remains somewhat heuristic. We have seen that some sequences of λ can take us to a non-global local minimum. Of course the fact that we start with a global minimum for $\lambda = 0$ is of some help, but we do not know how far it will take us in general. Jumps near λ_+ may indicate bifurcations to other local minima.

We have not stressed in the paper that minimizing the penalty function is a *continuation method* (Allgower and George (1979)). This means that probably better methods are available to follow the trajectory of solutions along $\lambda > 0$. There are also possibilities in exploring the fact that the maximum over Z of the penalty function (9) is a concave function of the single variable λ, which is a constant function for all $\lambda > \lambda_+$. There is a duality theory associated with these Courant penalty functions, which we have not used or explored so far.

Appendix A: Exterior Penalty Methods

Suppose $\mathcal{X} \subseteq \mathbb{R}^n$ and $f : \mathbb{R}^n \Rightarrow \mathbb{R}$ is continuous. Define

$$\mathcal{X}_\star = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\ f(x)$$

Suppose $\mathcal{X}_\star$ is non-empty and that $x_\star$ is any element of $\mathcal{X}_\star$, and

$$f_\star = f(x_\star) = \min_{x \in \mathcal{X}}\ f(x).$$

The following convergence analysis of external linear penalty methods is standard and can be found in many texts (for example, Zangwill (1969), section 12.2).

The penalty term $g : \mathbb{R}^n \Rightarrow \mathbb{R}^+$ is continuous and satisfies $g(x) = 0$ if and only if $x \in \mathcal{X}$. For each $\lambda > 0$ we define the (linear, external) penalty function

$$h(x, \lambda) = f(x) + \lambda g(x). \tag{15}$$

Suppose $\{\lambda_k\}$ is a strictly increasing sequence of positive real numbers. Define

$$\mathcal{X}_k = \underset{x\in\mathcal{X}}{\operatorname{argmin}}\ h(x,\lambda_k). \tag{16}$$

Suppose all $\mathcal{X}_k$ are nonempty and contained in a compact subset of $\mathcal{X}$. Choose $x_k \in \mathcal{X}_k$ arbitrarily.

Lemma 2: [Basic]

1: $h(x_k,\lambda_k) \leq h(x_{k+1},\lambda_{k+1})$.

2: $g(x_k) \geq g(x_{k+1})$.

3: $f(x_k) \leq f(x_{k+1})$.

4: $f_\star \geq h(x_k,\lambda_k) \geq f(x_k)$.

Proof:

1: We have the chain

$$h(x_{k+1},\lambda_{k+1}) = f(x_{k+1})+\lambda_{k+1}g(x_{k+1}) \geq f(x_{k+1})+\lambda_k g(x_{k+1}) \geq f(x_k)+\lambda_k g(x_k) = h(x_k,\lambda_k).$$

2: Both

$$f(x_k)+\lambda_k g(x_k) \leq f(x_{k+1})+\lambda_k g(x_{k+1}), \tag{17}$$
$$f(x_{k+1})+\lambda_{k+1}g(x_{k+1}) \leq f(x_k)+\lambda_{k+1}g(x_k). \tag{18}$$

Adding inequalities (17) and (18) gives

$$\lambda_k g(x_k)+\lambda_{k+1}g(x_{k+1}) \leq \lambda_k g(x_{k+1})+\lambda_{k+1}g(x_k),$$

or

$$(\lambda_k-\lambda_{k+1})g(x_k) \leq (\lambda_k-\lambda_{k+1})g(x_{k+1}),$$

and thus $g(x_k) \geq g(x_{k+1})$.

3: First

$$f(x_{k+1})+\lambda_k g(x_{k+1}) \geq f(x_k)+\lambda_k g(x_k). \tag{19}$$

We just proved that $g(x_{k+1}) \geq g(x_k)$, and thus

$$f(x_k)+\lambda_k g(x_k) \geq f(x_k)+\lambda_k g(x_{k+1}). \tag{20}$$

Combining inequalities (19) and (20) gives $f(x_{k+1}) \geq f(x_k)$.

4: We have the chain

$$f_\star = f(x_\star)+\lambda_k g(x_\star) \geq f(x_k)+\lambda_k g(x_k) \geq f(x_k).$$

■

Theorem 3: Suppose the sequence $\{\lambda_k\}_{k\in K}$ diverges to ∞ and $x_{\star\star}$ is the limit of any convergent subsequence $\{x_\ell\}_{\ell\in L}$. Then $x_{\star\star}\in\mathcal{X}_\star$, and $f(x_{\star\star}) = f_\star$, and $g(x_{\star\star}) = 0$.

Proof: Using part 4 of lemma 2

$$\lim_{\ell \in L} h(x_\ell, \lambda_\ell) = \lim_{\ell \in L} \{f(x_\ell) + \lambda_\ell g(x_\ell)\} = f(x_{\star\star}) + \lim_{\ell \in L} \lambda_\ell g(x_\ell) \leq f(x_\star).$$

Thus $\{h(x_\ell, \lambda_\ell)_{\ell \in L}\}$ is a bounded increasing sequence, which consequently converges, and $\lim_{\ell \in L} \lambda_\ell g(x_\ell)$ also converges. Since $\{\lambda_\ell\}_{\ell \in L} \rightarrow \infty$ it follows that $\lim_{\ell \in L} g(x_\ell) = g(x_{\star\star}) = 0$. Thus $x_{\star\star} \in \mathcal{X}$. Since $f(x_\ell) \leq f_\star$ we see that $f(x_{\star\star}) \leq f_\star$, and thus $x_{\star\star} \in \mathcal{X}_\star$ and $f(x_{\star\star}) = f_\star$.
■

Appendix B: Code

penalty.R

```r
smacofLoss <- function (d, w, delta) {
  return (sum (w * (delta - d) ^ 2) / 4)
}

smacofBmat <- function (d, w, delta) {
  dd <- ifelse (d == 0, 0, 1 / d)
  b <- -dd * w * delta
  diag (b) <- -rowSums (b)
  return(b)
}

smacofVmat <- function (w) {
  v <- -w
  diag(v) <- -rowSums(v)
  return (v)
}

smacofGuttman <- function (x, b, vinv) {
  return (vinv %*% b %*% x)
}

columnCenter <- function (x) {
  return (apply (x, 2, function (z) z - mean (z)))
}

smacofComplement <- function (y, v) {
  return (sum (v * tcrossprod (y)) / 4)
}

smacofPenalty <-
  function (w,
            delta,
            p = 2,
            lbd = 0,
            zold = columnCenter (diag (nrow (delta))),
            itmax = 10000,
            eps = 1e-10,
            verbose = FALSE) {
    itel <- 1
    n <- nrow (zold)
    vmat <- smacofVmat (w)
    vinv <- solve (vmat + (1 / n)) - (1 / n)
    dold <- as.matrix (dist (zold))
    mold <- sum (w * delta * dold) / sum (w * dold * dold)
    zold <- zold * mold
    dold <- dold * mold
    yold <- zold [, (p + 1) : n]
    sold <- smacofLoss (dold, w, delta)
    bold <- smacofBmat (dold, w, delta)
    told <- smacofComplement (yold, vmat)
    uold <- sold + lbd * told
    repeat {
      znew <- smacofGuttman (zold, bold, vinv)
      ynew <- znew [, (p + 1) : n] / (1 + lbd)
      znew [, (p + 1) : n] <- ynew
      xnew <- znew [, 1:p]
      dnew <- as.matrix (dist (znew))
      bnew <- smacofBmat (dnew, w, delta)
      tnew <- smacofComplement (ynew, vmat)
      snew <- smacofLoss (dnew, w, delta)
      unew <- snew + lbd * tnew
      if (verbose) {
        cat(
          "itel ",
          formatC(itel, width = 4, format = "d"),
          "sold ",
          formatC(
            sold,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "snew ",
          formatC(
            snew,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "told ",
          formatC(
            told,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "tnew ",
          formatC(
            tnew,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "uold ",
          formatC(
            uold,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "unew ",
          formatC(
            unew,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "\n"
        )
      }
      if (((uold - unew) < eps) || (itel == itmax)) {
        break
      }
      itel <- itel + 1
      zold <- znew
      bold <- bnew
      sold <- snew
      told <- tnew
      uold <- unew
```

```
    }
    zpri <-znew %*% svd(znew)$v
    xpri <- zpri[, 1:p]
    return (list (
      x = xpri,
      z = zpri,
      b = bnew,
      l = lbd,
      s = snew,
      t = tnew,
      itel = itel
    ))
  }
```

runPenalty.R

```
runPenalty <-
  function (w,
            delta,
            lbd,
            p = 2,
            itmax = 10000,
            eps = 1e-10,
            cut = 1e-6,
            write = TRUE,
            verbose = FALSE) {
    m <- length (lbd)
    hList <- as.list (1:m)
    hList[[1]] <-
      smacofPenalty(
        w,
        delta,
        p,
        lbd = lbd[1],
        itmax = itmax,
        eps = eps,
        verbose = verbose
      )
    for (j in 2:m) {
      hList[[j]] <-
        smacofPenalty(
          w,
          delta,
          p,
          zold = hList[[j - 1]]$z,
          lbd = lbd[j],
          itmax = itmax,
          eps = eps,
          verbose = verbose
        )
    }
    mm <- m
    for (i in 1:m) {
      if (write) {
        cat(
          "itel",
          formatC(hList[[i]]$itel, width = 4, format = "d"),
          "lambda",
          formatC(
            hList[[i]]$l,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "stress",
          formatC(
            hList[[i]]$s,
            width = 8,
            digits = 6,
            format = "f"
          ),
          "penalty",
          formatC(
            hList[[i]]$t,
            width = 8,
            digits = 6,
            format = "f"
          ),
          "\n"
        )
      }
      if (hList[[i]]$t < cut) {
        mm <- i
        break
      }
    }
    return(hList[1:mm])
  }
```

```r
writeSelected <- function(hList, ind) {
  m <- length(hList)
  n <- length(ind)
  mn <- sort(union(union(1:3, ind), m - (2:0)))
  for (i in mn) {
    if (i > m) {
      next
    }
    cat(
      "itel",
      formatC(hList[[i]]$itel, width = 4, format = "d"),
      "lambda",
      formatC(
        hList[[i]]$l,
        width = 10,
        digits = 6,
        format = "f"
      ),
      "stress",
      formatC(
        hList[[i]]$s,
        width = 8,
        digits = 6,
        format = "f"
      ),
      "penalty",
      formatC(
        hList[[i]]$t,
        width = 8,
        digits = 6,
        format = "f"
      ),
      "\n"
    )
  }
}
```

matchMe.R

```r
matchMe <- function (x,
                     itmax = 100,
                     eps = 1e-10,
                     verbose = FALSE) {
```

```
  m <- length (x)
  y <- sumList (x) / m
  itel <- 1
  fold <- sum (sapply (x, function (z)
    (z - y) ^ 2))
  repeat {
    for (j in 1:m) {
      u <- crossprod (x[[j]], y)
      s <- svd (u)
      r <- tcrossprod (s$u, s$v)
      x[[j]] <- x[[j]] %*% r
    }
    y <- sumList (x) / m
    fnew <- sum (sapply (x, function (z)
      (z - y) ^ 2))
    if (verbose) {

    }
    if (((fold - fnew) < eps) || (itel == itmax))
      break
    itel <- itel + 1
    fold <- fnew
  }
  return (x)
}

sumList <- function (x) {
  m <- length (x)
  y <- x[[1]]
  for (j in 2:m) {
    y <- y + x[[j]]
  }
  return (y)
}
```

plotMe.R

```
plotMe2 <- function(hList, labels, s = 1, t = 2) {
  n <- nrow(hList[[1]]$x)
  m <- length (hList)
  par(pty = "s")
  hMatch <- matchMe (lapply (hList, function(r)
    r$x))
  hMat <- matrix (0, 0, 2)
  for (j in 1:m) {
    hMat <- rbind(hMat, hMatch[[j]][, c(s, t)])
  }
  plot(hMat,
       xlab = "dim 1",
       ylab = "dim 2",
       col = c(rep("RED", n*(m-1)), rep("BLUE", n)),
       cex = c(rep(1, n*(m-1)), rep(2, n)))
  for (i in 1:n) {
    hLine <- matrix (0, 0, 2)
    for (j in 1:m) {
      hLine <- rbind (hLine, hMatch[[j]][i, c(s, t)])
    }
    lines(hLine)
  }
  text(hMatch[[m]], labels, cex = .75)
}

plotMe1 <- function(hList, labels) {
  n <- length (hList[[1]]$x)
  m <- length (hList)
  blow <- function (x) {
    n <- length (x)
    return (matrix (c(1:n, x), n, 2))
  }
  hMat <- matrix (0, 0, 2)
  for (j in 1:m) {
    hMat <- rbind(hMat, blow(hList[[j]]$x))
  }
  plot(hMat,
       xlab = "index",
       ylab = "x",
       col = c(rep("RED", n*(m-1)), rep("BLUE", n)),
       cex = c(rep(1, n*(m-1)), rep(2, n)))
  for (i in 1:n) {
    hLine <- matrix (0, 0, 2)
    for (j in 1:m) {
      hLine <- rbind (hLine, blow(hList[[j]]$x)[i, ])
      lines(hLine)
    }
  }
  text(blow(hList[[m]]$x), labels, cex = 1.00)
  for (i in 1:n) {
    abline(h = hList[[m]]$x[i])
  }
}
```

checkUni.R

```
checkUni <- function (w, delta, x) {
  x <- drop (x)
  n <- length (x)
  vinv <- solve (smacofVmat (w) + (1 / n)) - (1 / n)
  return (drop (vinv %*% rowSums (w * delta * sign (outer (x, x, "-")))))
}
```

References

Allgower, E. L., and K. George. 1979. *Introduction to Numerical Continuation Methods*. Wiley.

Borg, I., and P. J. F. Groenen. 2005. *Modern Multidimensional Scaling*. Second Edition. Springer.

Cox, D. R., and L. Brandwood. 1959. "On a Discriminatory Problem Connected with the Works of Plato." *Journal of the Royal Statistical Society, Series B* 21: 195–200.

De Gruijter, D. N. M. 1967. "The Cognitive Structure of Dutch Political Parties in 1966." Report E019-67. Psychological Institute, University of Leiden.

De Leeuw, J. 1977. "Applications of Convex Analysis to Multidimensional Scaling." In *Recent Developments in Statistics*, edited by J. R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, 133–45. Amsterdam, The Netherlands: North Holland Publishing Company.

———. 1984. "Differentiability of Kruskal's Stress at a Local Minimum." *Psychometrika* 49: 111–13.

———. 1993. "Fitting Distances by Least Squares." Preprint Series 130. Los Angeles, CA: UCLA Department of Statistics. https://jansweb.netlify.app/publication/deleeuw-r-93-c/deleeuw-r-93-c.pdf.

———. 2005. "Unidimensional Scaling." In *The Encyclopedia of Statistics in Behavioral Science*, edited by B. S. Everitt and D. Howell, 4:2095–97. New York, N.Y.: Wiley.

———. 2014. "Bounding, and Sometimes Finding, the Global Minimum in Multidimensional Scaling." UCLA Department of Statistics.

———. 2016. "Gower Rank." 2016.

———. 2017. "Shepard Non-metric Multidimensional Scaling." 2017.

De Leeuw, J., and P. Mair. 2009. "Multidimensional Scaling Using Majorization: SMACOF in R." *Journal of Statistical Software* 31 (3): 1–30.

Dolgopolik, M. V. 2016a. "A Unifying Theory of Exactness of Linear Penalty Functions." *Optimization* 65 (6): 1167–1202.

———. 2016b. "Smooth Exact Penalty Functions: a General Approach." *Optimization Letters* 10: 635–48.

———. 2017. "A Unifying Theory of Exactness of Linear Penalty Functions II: Parametric Penalty Functions." *Optimization* 66 (10): 1577–1622.

———. n.d. "Smooth Exact Penalty Functions: a General Approach."

Ekman, G. 1954. "Dimensions of Color Vision." *Journal of Psychology* 38: 467–74.

Guilford, J. P. 1954. *Psychometric Methods.* McGraw-Hill.

Mair, P., P. J. F. Groenen, and J. De Leeuw. 2022. "More on Multidimensional Scaling in R: smacof Version 2." *Journal of Statistical Software* 102 (10): 1–47.

Palubeckis, G. 2013. "An Improved Exact Algorithm for Least-Squares Unidimensional Scaling." *Journal of Applied Mathematics*, 1–15.

Revelle, W. 2018. *psych: Procedures for Psychological, Psychometric, and Personality Research.* Evanston, Illinois: Northwestern University.

Rockafellar, R. T. 1970. *Convex Analysis.* Princeton University Press.

Rothkopf, E. Z. 1957. "A Measure of Stimulus Similarity and Errors in some Paired-associate Learning." *Journal of Experimental Psychology* 53: 94–101.

Shepard, R. N. 1962a. "The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. I." *Psychometrika* 27: 125–40.

———. 1962b. "The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. II." *Psychometrika* 27: 219–46.

Zangwill, W. I. 1969. *Nonlinear Programming: a Unified Approach.* Englewood-Cliffs, N.J.: Prentice-Hall.