ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2007, Vol.36, No.4

DIAGONAL MAJORIZATION ALGORITHM: PROPERTIES AND EFFICIENCY

Jolita Bernatavičienė [1,2], Gintautas Dzemyda [1,2], Virginijus Marcinkevičius [1]

[1] Institute of Mathematics and Informatics
Akademijos St. 4, LT-08663, Vilnius, Lithuania
[2] Vilnius Pedagogical University
Studentų St. 39, LT-08106, Vilnius, Lithuania

Abstract. In this paper, the diagonal majorization algorithm (DMA) has been investigated. The research focuses on the possibilities to increase the efficiency of the algorithm by disclosing its properties. The diagonal majorization algorithm is oriented at the multidimensional data visualization. The experiments have proved that, when visualizing large data set with DMA, it is possible to save the computing time taking into account several factors: the strategy of numbering of multidimensional vectors in the analysed data set and the neighbourhood order parameter.

Keywords: multidimensional scaling, visualization, large data set, diagonal majorization algorithm.

1. Introduction

At present, computer systems store large amounts of data. Data from the real world are often described by an array of parameters, i.e., we deal with multidimensional data. It is much easier for a human to observe, detect, or extract some information from the graphical representation of these data (to detect the presence of clusters, outliers or various regularities in the analysed data) than from raw numbers.

One of the most popular methods for graphical representation (visualization) of multidimensional data is multidimensional scaling (MDS) [5]. It found a lot of applications in technology [12], economy, medicine [1], psychology [9], etc.

We face the problems with the standard MDS algorithm when we have to project (visualize) a large data set, or a new data point among the previously mapped points has to be projected. In the standard MDS, every iteration requires each point to be compared with all other points and the iteration complexity is $O(m^2)$, here m is the number of vectors in analysed data set. Thus, the MDS method is unsuitable for large data sets: it takes much computing time or there is not enough computing memory. Various modifications of MDS have been proposed to visualize of large data sets: Steerable multidimensional scaling (MDSteer) [16], Incremental MDS, Relative MDS [3, 13], Landmark MDS [14], Diagonal majorization algorithm (DMA) [15], etc.

This paper focuses on the diagonal majorization algorithm and on the possibilities to increase its efficiency by disclosing its properties.

2. Background for the diagonal majorization algorithm

Multidimensional scaling (MDS) is a group of methods that project multidimensional data to a low- (usually two-) dimensional space and preserve the interpoint distances among data as much as possible. Let us have vectors (points) $X^i = (x_1^i, x_2^i, ..., x_n^i)$, $i = 1,...,m$ ($X^i \in R^n$). The pending problem is to get the projection of these n-dimensional vectors X^i, $i = 1,...,m$ on the plane R^2. The two-dimensional vectors $Y^1, Y^2, ..., Y^m \in R^2$ correspond to them. Here $Y^i = (y_1^i, y_2^i)$, $i = 1,...,m$. Denote the distance between the vectors X^i and X^j by d_{ij}^*, and the distance between the corresponding vectors on the projected space (Y^i and Y^j) by d_{ij}. In our case, the initial dimensionality is n, and the resulting one is 2. There exists a multitude of variants of MDS with slightly different so-called stress functions. In our experiments, the raw stress (projection error) is minimized:

$$E_{MDS} = \sum_{\substack{i,j=1 \\ i<j}}^{m} w_{ij} (d_{ij}^* - d_{ij})^2 \qquad (1)$$

where w_{ij} are weights.

Various types of minimization of the stress function are possible (see [5], [11]). In this paper, we use the Guttman majorization algorithm (GMA), based on iterative majorization and its modification so-called diagonal majorization algorithm (DMA) [15]. GMA is one of the best optimisation algorithms for this type of minimization problem [5], [6]. This method is simple and powerful, because it guarantees a monotone convergence of the stress function.

Formula (2) is called the Guttman transform [7].

$$Y(t'+1) = V^{\dagger} B(Y(t'))Y(t'), \qquad (2)$$

where $B(Y(t'))$ has the entries:

$$b_{ij} = \begin{cases} -\dfrac{d_{ij}^*}{d_{ij}}, & \text{for } i \neq j \text{ and } d_{ij} \neq 0 \\ 0, & \text{for } i \neq j \text{ and } d_{ij} = 0 \end{cases},$$

$$b_{ii} = -\sum_{j=1,\, j\neq i}^{m} b_{ij} .$$

V is a matrix of weights where V has the entries:

$$V = \begin{pmatrix} \sum w_{1s} & & & & \\ & \ddots & & -w_{ij} & \\ & & \ddots & & \\ & -w_{ij} & & \ddots & \\ & & & & \sum w_{ms} \end{pmatrix}. \qquad (3)$$

$V^{\dagger}$ denotes the Moore-Penrose pseudoinverse of V.

In our case w_{ij} =1, therefore $V^{\dagger} = \frac{1}{m} I$, here I is an identity matrix.

This algorithm for MDS can be summarized as follows (see details in [5]):

1. Set the initial values of Y, set $t' = 0$.
2. Compute the value of the stress function $E_{MDS}(Y(t'))$ by (1); here the two-dimensional vectors Y are set in $t' = 0$.
3. Increase the iteration counter t' by one.
4. Compute the Guttman transform $Y(t')$ by (2).
5. Compute $E_{MDS}(Y(t'))$.
6. If $E_{MDS}(Y(t'-1)) - E_{MDS}(Y(t')) < \varepsilon$ or t' is equal to the maximum number of iterations, then stop (ε is a small positive constant), else go to Step 3.

Diagonal majorization algorithm (DMA) was proposed in [15]. DMA uses simpler majorization function:

$$Y(t'+1) = Y(t') + \frac{1}{2} diag(V)^{-1} [B(Y(t') - V)]Y(t'). \quad (4)$$

DMA attains slightly worse projection error than GMA, but computing by the iteration equation (4) is faster and there is no need of computing the pseudoinverse $V^{\dagger}$ matrix. In addition, DMA differs from GMA that a large number of entries b_{ij} in matrix V have zero values. This means that iterative computations of two-dimensional coordinates $Y^i = (y_1^i, y_2^i)$, $i = 1,\ldots,m$ are based not on all distances d_{ij}^* between multidimensional points X^i and X^j. This allows to speed up the visualization process and to save the computer memory essentially.

This algorithm, however, remains of $O(m^2)$ complexity if we use the standard V matrix (3). With a view to diminish the complexity of this algorithm, Trosset and Groenen [15] recommended to use only a part weights of matrix V and propose two ways of forming the matrix V:

1. The weights are defined by setting $w_{ij} = 0.4/(0.4 + d_{ij}^*)$ for $d_{ij}^* < 0.6$ and $w_{ij} = 0$, otherwise. Here $d_{ij}^* < 0.6$ defines some boundary. We do not use this way in our experiments, because it is difficult to estimate the number of distances d_{ij}^* that will be taken into consideration, and the sufficient distance boundary for the analysed data set.
2. The weights are defined by setting $w_{ij} = 1$ for k "cycles" of d_{ij}^*, e.g., $i \leftrightarrow i \pm 1, \ldots, i \leftrightarrow i \pm k$, etc. and $w_{ij} = 0$, otherwise. The parameter k defines neighbourhood order for point X^i in the list of analysed data set vectors. In this case, the complexity of DMA algorithm decreases till $O(km)$.

Trosset and Groenen [15] however have not determined how much the result depends on selection of the parameter k.

To compare the obtained visualization results, the projection error is calculated:

$$E = \sqrt{\sum_{i<j}^{m} (d_{ij}^* - d_{ij})^2 \Big/ \sum_{i<j}^{m} (d_{ij}^*)^2} . \qquad (5)$$

The projection error E (5) is used here instead of E_{MDS} (1), because the inclusion of the normalized parameter $\sum_{i<j}^{m} (d_{ij}^*)^2$ gives a clear interpretation of the image quality that does not depend on the scale and the number of distances in an n-dimensional space.

3. Data sets for analysis

Five data sets were used in our experiments. Four of them are artificial, and the last one is real, namely:

1. *Ellipsoidal*[*m, n*] set, m=3140, n=50; the set contains 10 overlapping ellipsoidal-type clusters. In the experiments, data set, obtained using the ellipsoidal cluster generator [8], is used. This generator creates ellipsoidal clusters with the major axis of an arbitrary orientation. The boundary of a cluster is defined by four parameters:
   - the origin (which is also the first focus);
   - the interfocal distance, uniformly distributed in the range [1.0, 3.0];
   - the orientation of the major axis, uniformly located amongst all orientations;
   - the maximum sum of Euclidean distances to two foci, belonging to the range [1.05, 1.15] – equivalent to the eccentricity ranging from [0.870, 0.952].

   For each cluster, data points are generated at a Gaussian-distributed distance from a uniformly random point on the major axis, in a uniformly random direction, and are rejected if they lie outside the boundary. Using this ellipsoidal generator, an *Ellipsoidal* data set is generated.
2. *Gaussian*[*m*, *n*] set, where m=2729, n=10; it contains 10 overlapping clusters. The Gaussian data set has been generated, using the Gaussian cluster generator. This generator is based on a standard cluster model using multivariate normal distributions. See [8] for more details.
3. *Paraboloid*[*m, n*] set, where m=2583, n=3; the data set contains 2 non-overlapping clusters. The *Paraboloid* data set is also an artificially created data set. There are two classes in this data set. The vectors are generated as follows: the first two coordinates of the vector are randomly generated in a predefined area (for the first class it is a circle with radius 0.4, for the second class this area is a ring, limited by two circles with radii 0.7 and 1.2). Then the third coordinate is added using such a rule $x_3 = 1.8 \cdot \sqrt{(x_1)^2 + (x_2)^2}$. The created paraboloid is rotated to make the classification more difficult.
4. *Abalone*[*m, n*] set, where m=4177, n=8 which contains 29 clusters. This data set is taken from the UCI repository [4]. Each vector is described by 8 parameters of abalone: x_1 – length (the longest shell measurement), x_2 – diameter (perpendicular to the length), x_3 – height (with meat in the shell), x_4 – the whole weight of abalone, x_5 – shucked weight (the weight of meat), x_6 – viscera weight (gut weight after bleeding), x_7 – shell weight after being dried, and x_8 – rings. The data set samples are highly overlapping. Since the scales of parameters are different, it is necessary to normalize them: to calculate the average $\overline{x_j}$ and variance σ_j^2 of each parameter; the values of each parameter x_{ij} are normalized by the formula: $(x_{ij} - \overline{x}_j)/\sigma_j$.

4. Selection of the parameter *k*

As it is mentioned above, parameter k defines neighbourhood order for point X^i in the list of analysed data set vectors.

Selection of the parameter k in the diagonal majorization algorithm (DMA) has a great influence on the projection error (5) and obtained map. It has been investigated how the projection error is varying by increasing the parameter k, the computing time and number of iterations being fixed. The vectors of the initial analysed data set were mixed at random in each experiment so that there were less similar points in the list of analysed data points. Having done 50 experiments for each k, when k varied from 100 to 1000, by step 100, the averages of errors were computed. The initial two-dimensional vectors were initiated in GMA and DMA algorithms by the method of principal component analysis (PCA) [2], [10].

The experiment has shown (see Figures 1 and 2) that, for $k \geq 400$ and under the fixed computing time, already after 300 iterations one can get quite an accurate result. Projection error increases less than 1 % comparing with GMA algorithm. With an increase in the number of iterations, the error changes but slightly. By increasing k considerably, the computing time also increases, while the result approaches that obtained by GMA algorithm ($E = 0.043497$ for *abalone* data and $E = 0.210109$ for *ellipsoidal* data).

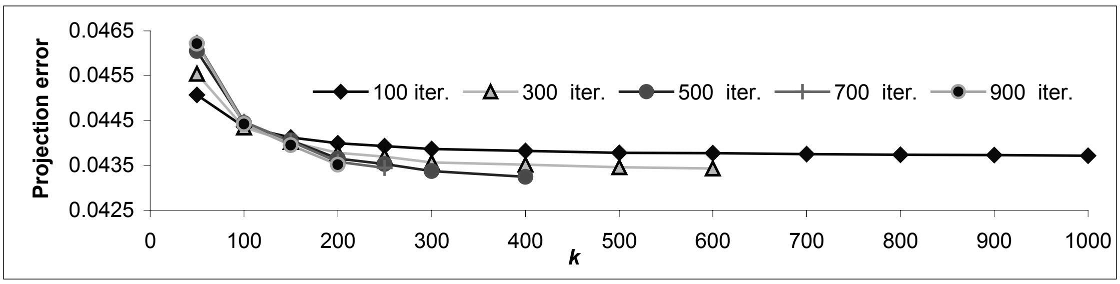

Figure 1. Dependence of the projection error on the neighbourhood order parameter *k* (for *abalone* data set)

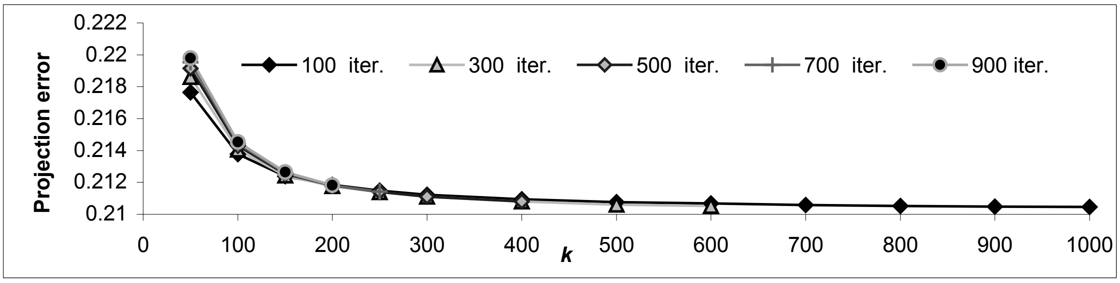

Figure 2. Dependence of the projection error on the neighbourhood order parameter *k* (for *ellipsoidal* data set)

This experiment has also illustrated that for too small *k* increasing number of iterations, in many cases the error does not decrease but, vice verse, increases (see Figures 1 and 2, $k = 50$).

5. Results of comparative analysis

Carrying out the experiments with different data sets, it has been established that the projection error is influenced a great deal by formation of set of multidimensional points, i.e., numbering of vectors in analysed data set. To corroborate this fact, the following investigation was performed. The initial set of multidimensional data was made up using three different strategies of points numbering of the set:

1. (Strategy I). At the beginning of algorithm operation, the points of analysed multidimensional data set are mixed up at random (one random numbering).
2. (Strategy II). The points of multidimensional data set and two-dimensional vectors, corresponding to these multidimensional points, and whose coordinates have been calculated in the previous iterations, are randomly mixed up in the operation of the algorithm at the beginning of the each iteration (random numbering before each iteration).
3. (Strategy III). Using the method of the principal component analysis (PCA) [10], multidimensional vectors are projected onto a straight line, thus establishing the similarity of this point, and multidimensional data are numbered in this order (closer points should have similar order numbers).

Using Strategies I and II for multidimensional vector numbering, 50 experiments have been done with each *k*, varying it from 50 to 1000, the data have been visualized, the averages of projection error and standard deviation as well as computing time has been recorded. Since the previous experiments have shown that the error changes insignificantly after more than 300 iterations, the algorithms have been iterated 300 times each in this experiment (Strategy I is used, Figures 1 and 2).

Using Strategy II, even after 100 iterations rather good results have been obtained and by increasing the number of iterations they almost do not change. Using this strategy, the least error is obtained, when these three strategies were compared. The projection error varies insignificantly by increasing *k* (Figures 3, 4, and 5).

Increasing parameter *k* from 300 to 1000, the projection error decreases by the rule $E = -0.0002\ln(k) + C$, here C is a constant. It means that in this case, the projection error will be decreased till 0.08% approximately.

The experiments done have illustrated that numbering of multidimensional data (Strategy III) worsens the visualizations results (Figures 3 and 4). If we employ the DMA algorithm, we need close and distant points side by side, because taking them into consideration the coordinates of two-dimensional vectors are computed. Mixing of multidimensional vectors at each iteration implies that when calculating the coordinates of a two-dimensional point, more and various neighbors are regarded, which results in a more accurate projection (Strategy II) and it suffices less iterations (100 is enough) (Figure 5).

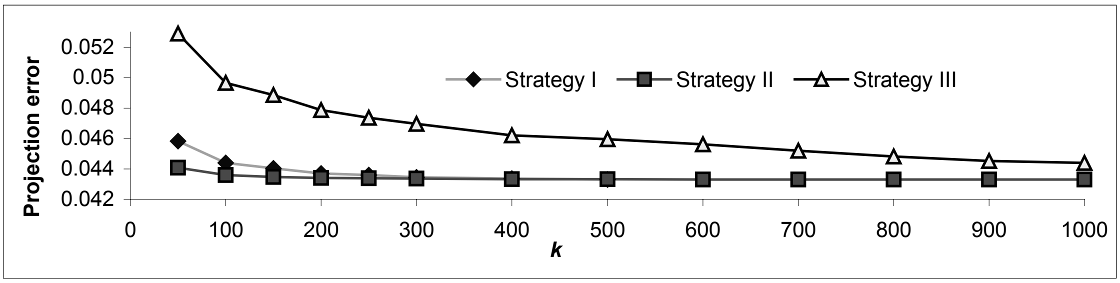

Figure 3. Dependence of the projection error on the neighbourhood order parameter *k* (for *abalone* data set), using different numbering strategies

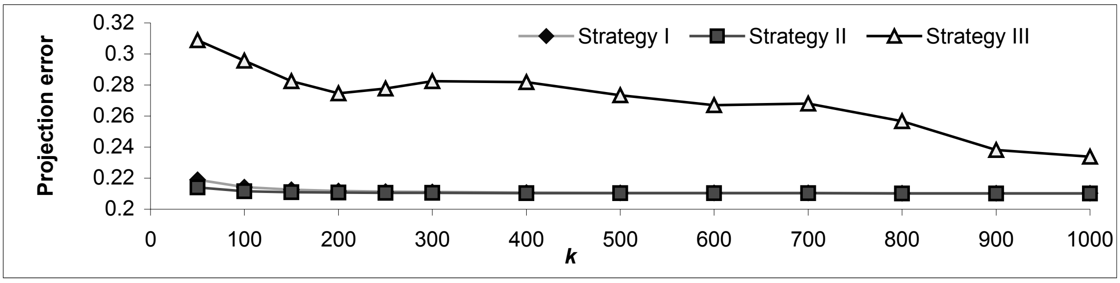

Figure 4. Dependence of the projection error on the neighbourhood order parameter k (for *ellipsoidal* data set), using different numbering strategies

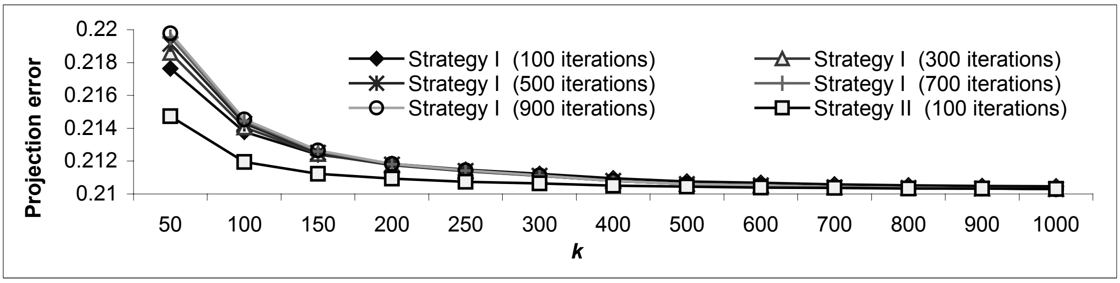

Figure 5. Dependence of the projection error on the neighbourhood order parameter k (*ellipsoidal* data set), using different numbering strategies (Strategies I and II) and different number of iterations

Table 1. Projection error, standart deviation of the projection error and computing time for both algorithm

Data	GMA(100 iter.)		DMA (Strategy I, 300 iter., k=400)			DMA (Strategy II, 100 iter., k=400)		
	Error	time, s	Average of Error	St. deviation of Error	time, s	Average of Error	St. deviation of Error	time, s
abalone	0.043497	151.82	0.043493	0.000065	29.10	0.043726	0.000015	21.96
paraboloid	0.208653	22.40	0.217755	0.004121	16.98	0.214324	0.002012	9.74
ellipsoidal	0.210109	105.27	0.210794	0.000145	23.44	0.210501	0.000059	15.81
gaussian	0.283866	109.28	0.285405	0.000290	28.60	0.284212	0.000039	19.10

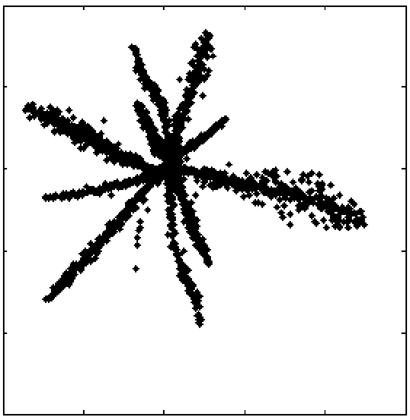

Figure 6. Projection of the *ellipsoidal* data obtained by GMA algorithm: t=94.22s, E=0.210109

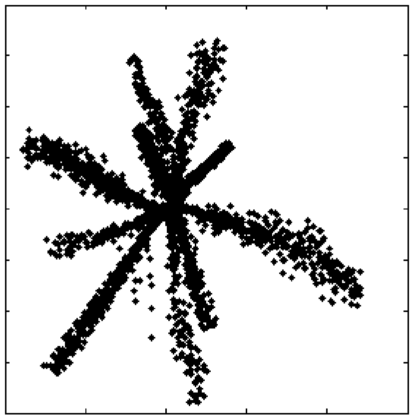

Figure 7. Projection of the *ellipsoidal* data obtained by DMA algorithm: k=400, t=13.84s, E=0.210848

Also the GMA and DMA algorithms have been compared with respect to time and projection error. The results are presented in Table 1. After the experiments with four different data sets, it has been established that the projection error, obtained by GMA, is slightly smaller, while using DMA, the computing time is considerably shorter. The larger the set, the more distinct the computing time difference is. By comparing visualization results obtained by GMA and DMA, we notice no great difference between the obtained projections, since the difference between errors is very little ($\leq 1\%$ for *abalone*, *gaussian* and *ellipsoidal* data sets, and $\leq 4\%$ for *paraboloid* data set). Figures 6 and 7 present the *ellipsoidal* data projections, obtained by GMA and DMA. 100 iterations have been performed in both cases. The DMA employed Strategy II for numbering of a multidimensional data set. In both Figures 6 and 7 the data structure is sufficiently clear, though errors are slightly different. However, the difference between computing times is distinct, the projection has been obtained by DMA 7 times quicker. This difference of computing time decreases by increasing the amount of vectors in the analysed data set and decreasing parameter k, because data preprocessing for iteration process, using Strategy II, requires more calculations.

6. Conclusions

The experiments in this paper have proved that when visualizing large data set, the diagonal majorization algorithm (DMA) is efficient in saving the computing time when a good visualization quality is required, however, this is influenced by several factors: the strategy of numbering of multidimensional vectors in the analysed data set and the neighbourhood order parameter k. The projection error by DMA is a little worse than that obtained by GMA, however, by selecting the parameter $k \geq 400$, rather low error is obtained.

We have established the dependence of DMA efficiency on the numbering of points of the analysed set, therefore several strategies of numbering are offered, the usage of which resulted in low visualization errors for the smaller neighbourhood order parameter k and has saved time considerably.

Renumbering of multidimensional points of the analysed data set before each iteration has turned to be particularly efficient, because, in this case, almost all the points of the analysed set actually take part in the calculation of each multidimensional point projection.

Acknowledgment

The research is partially supported by the Lithuanian State Science and Studies Foundation project "Information technology tools of clinical decision support and citizens wellness for e.Health system (No. B-07019)".

References

[1] **J. Bernatavičienė, G. Dzemyda, O. Kurasova, V. Marcinkevičius, V. Medvedev.** The problem of visual analysis of multidimensional medical data. *Springer optimization and its applications, Models and algorithms for global optimization*, New York : Springer, ISBN 0-387-36720-9,*Vol*.4, 2007, 277–298.

[2] **J. Bernatavičienė, G. Dzemyda, O. Kurasova, V. Marcinkevičius.** Optimal decisions in combining the SOM with nonlinear projection methods. *European Journal of Operational Research*, *Vol*.173, 2006, 729–745.

[3] **J. Bernatavičienė, G. Dzemyda, V. Marcinkevičius.** Conditions for Optimal Efficiency of Relative MDS. *Informatica*. ISSN 0868-4952. *Vol*.18, *No*.2, 2007, 187–202.

[4] **C.L. Blake, C.J. Mertz.** UCI Repository of machine learning databases. *Irvine, CA: University of California, Department of Information and Computer Science*,1998.

[5] **I. Borg, P. Groenen.** Modern Multidimensional Scaling: Theory and Applications. *Springer, New York*, 1997.

[6] **P.J.F. Groenen, M. van de Vaelden.** Multidimensional Scaling, *Econometric Institute Report EI2004-15*, 2004, *https://ep.eur.nl/handle/1765/1274/1/ei200415.pdf.*

[7] **L. Guttman.** A general nonmetric technique for finding the smallest coordinate space for a configuration of points. *Psychometrika*, *Vol*.33, 1968, 469–506.

[8] **J. Handl, J. Knowles.** Cluster generators for large high-dimensional data sets with large numbers of clusters. *http://dbkgroup.org/handl/generators/.*

[9] **J.B. Kruskal.** Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, *Vol*.29, *No*.1, 1964, 1–27.

[10] **D.N Lawley, A.E. Maxwell.** Factor Analysis as a Statistical Method. *London, Butterworths*, 1963.

[11] **R. Mathar, A. Žilinskas.** On Global Optimization in Two-Dimensional Scaling, *Acta Aplicandae Mathematicae, Vol.* 33, 1993, 109–118.

[12] **A. Morrison, G. Ross, and M. Chalmers.** Fast multidimensional scaling through sampling, springs and interpolation. *Information Visualization*, *Vol*.2, *No*.1, 2003, 68–77.

[13] **A. Naud.** Visualization of high-dimensional data using a association of multidimensional scaling to clustering. *Proceedings of the* 2004 *IEEE Conference on Cybernetics and Intelligent Systems*, *Vol*.1, 2004, 252–255.

[14] **V. de Silva, J.B. Tenenbaum.** Global versus local methods for nonlinear dimensionality reduction. *In S. Becker, S. Thrun,, K. Obermayer (Eds.), Advances in Neural Information Processing Systems, Vol.* 15, *MIT Press, Cambridge, MA*, 2003, 721–728.

[15] **M.W. Trosset, P.J.F. Groenen.** Multidimensional scaling algorithms appear. *In Computing Science and Statistics*, 2005, *CD-ROM*.

[16] **M. Williams, T. Munzner.** Steerable, Progressive Multidimensional Scaling. *Information Visualization*, 2004. *INFOVIS* 2004. *IEEE Symposium on* 10-12 *Oct*. 2004, 57–64.

Received September 2007.

DOI: 10.5755/j01.itc.36.4.11888