Structured Peer-to-Peer Networks through Distributed Nonmetric Multidimensional Scaling

Hauke Coltzau
Fernuniversität in Hagen
hauke.coltzau@fernuni-hagen.de

Abstract: Multidimensional Scaling (MDS) is a technique for dimensionality reduction widely used in the field of data-mining. For P2P data networks, it holds the potential to provide an intuitive way to browse and explore heterogeneous data distributed over the system. Current MDS approaches are dependent on a centralized instance (controller) and can therefore not be used in decentralized environments. In this article, a distributed algorithm for multidimensional scaling is discussed that does not need any such centralized control.

1 Introduction

This article aims to show that a structured Peer-to-Peer (P2P) network can be built by arranging peers in a 2-dimensional space in such a way that peers storing similar content are located in the same area. The structure of the resulting P2P system provides intuitive ways of navigation and browsing through it's content. Users can search and explore such a network without exactly having to know beforehand, what the result of their search will be. For additional support of orientation and navigation, dynamic maps of such a system could be generated in a distributed manner (see [CU12]), and thus even give an overview of the type of objects stored in the system..

The relations between the objects stored in a peer-to-peer system usually can only be described in a high-dimensional space. Therefore, some distributed dimensionality reduction is needed to obtain the required 2-dimensional projection. In data mining, the term *Multidimensional Scaling* (MDS) denotes a set of such techniques. But these techniques require an omniscient (global) view on all objects, for which the dimensionality reduction shall be done. Hence, to achieve the goal of building a Peer-to-Peer system based on MDS, a distributed MDS technique is needed, which does neither need any centralized instances nor a global view on all participating peers.

The structure of this article therefore is as follows: In section 2, an overview of multidimensional scaling is given, including existing approaches for parallel and distributed MDS. In section 3, an algorithm is proposed to perform distributed MDS and build a structured P2P-system on the resulting configuration. Section 4 shows, how the algorithm performs in an example network. The results gathered from this and other simulations are described in the final section.

2 Multidimensional Scaling

MDS aims to visualize high-dimensional data in a low-dimensional (2D or 3D) target space while preserving (dis-)similarities between the data points. The idea of multidimensional scaling goes back to Torgerson [Tor52] and has been generalized by Kruskal [Kru64a][Kru64b]. All current adaptions and also large-scale approaches like [CPH06] and [BBKY06] are still strongly based on these works, so that a description of the basic idea behind them seems appropriate in the context of this article.

2.1 General Approach for MDS

The input of any multidimensional scaling algorithm is a set N of $n = |N|$ objects, from which the pairwise dissimilarities $\delta_{i,j}$ for (almost[1]) all objects $i, j \in N$ are known. Often, these dissimilarities are given in form of a dissimilarity matrix Δ:

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,1} & \dots & \delta_{1,1} \\ \delta_{2,1} & \delta_{2,2} & \dots & \delta_{2,n} \\ \vdots & \vdots & & \vdots \\ \delta_{n,1} & \delta_{n,2} & \dots & \delta_{n,n} \end{pmatrix} \tag{1}$$

Approaches for executing an MDS can roughly be categorized as *metric* or *non-metric*. Metric approaches expect the dissimilarity values to be in a (ratio-scaled) metric, such that a scalar distance between any to objects can be specified. Non-metric approaches in contrast only expect a rank-order of dissimilarities and any rank-order preserving function to get a scalar representation of each dissimilarity. Hence, non-metric approaches can be seen as generalization of metric approaches.

A *configuration* X constitutes a projection of N into s-dimensional space (where s is very often chosen to be 2 or 3), i.e. each object is assigned to an s-dimensional vector x_i.

$$X = (x_1, x_2, \dots, x_n), x_i \in \mathbb{R}^s \tag{2}$$

The *distance* $d_{i,j}$ between two vectors x_i and x_j in s-dimensional space is commonly measured in euclidian metric, although in some cases, other Minkowski r-metrics as e.g. the Manhattan metric are used.

Especially for non-metric MDS, dissimilarities have to be transformed into values that the distances $d_{i,j}$ in target space can be compared to. These values are called *disparities* $\hat{\delta}_{i,j}$, and can be interpreted as target values for the distances in low-dimensional space. How the disparities are calculated from the dissimilarities, depends both on the goal of the MDS as well as the type of input data. In section 2.2, an example approach for finding the

[1] Kruskal considers missing dissimilarity values as weighted with 0.0, and simply proposes to omit such values in all calculations.

disparities in non-metric MDS is given. In metric MDS, the disparities often are identical to the dissimilarities (i.e. $\hat{d}_{i,j} = \delta_{i,j}$).

To evaluate the fitness of a configuration X as a function of the disparities, a cost function (referred to as *stress*) is used. This function iterates over all object pairs $i, j \in N$ and compares their disparity $\hat{d}_{i,j}$ to the distance $d_{i,j} = |x_i - x_j|$ of their assigned vectors x_i and x_j in X. The better the low-dimensional projection reflects the dissimilarities in high-dimensional space, the lower is the stress and the better the chosen configuration represents the original data. A commonly used stress function is given in section 2.2.

The core task of MDS now is to find a configuration X_{min}, which minimizes the stress. In metric MDS, the minimum stress is achieved, when all distances in target space are as close as possible to the disparities (and therefore to the dissimilarities). In non-metric MDS, the stress is minimized, when the rank order of the distances in target space is equal to the rank order of the disparities (and therefore the dissimilarities).

Due to the complexity of the stress function, the minimization can usually not be done in an analytical manner. Instead, usually the method of steepest decent (i.e. gradient descent), or the approach of majorizing the stress function with a less complex function (*Scaling by majorizing a complicated function*, SMACOF [dL77]) are used, the latter being only applicable in metric MDS.

Usually, when an optimal configuration shall be found, the MDS is executed several times (up to 100) with different random initial configurations to avoid local minima. The one configuration with the lowest stress value is assumed to be the global optimum.

2.2 Kruskal's Approach for Nonmetric Multidimensional Scaling

Since almost all current approaches for MDS go back to the work of Torgeson and Kruskal, and since Kruskal's non-metric MDS can be seen as generalization of metric MDS, a brief description of Kruskal's approach is given. The main properties of the approach are the following:

1. Disparities are calculated in such a way, that the rank order of distances in target space is as close as possible to the rank order of dissimilarities. This is achieved using monotonic regression.
2. A steepest decent is used to approximate the configuration step by step. To do so, at each iteration a gradient is calculated, which is applied to the configuration. The iteration stops, when the first derivative of the gradient is zero, i.e. when a minimum is reached.

2.2.1 Disparity Calculation

A good solution (configuration) of the non-metric MDS is found, when the configuration preserves the rank-order of the dissimilarities, i.e:

$$\delta_{i,j} < \delta_{j,k} < \delta_{k,l} < \ldots \iff d_{i,j} < d_{j,k} < d_{k,l} < \ldots \tag{3}$$

In other words: When the dissimilarities $\delta_{i,j}$ are sorted in ascending order and the distances $d_{i,j}$ are then plotted over the dissimilarities, the resulting curve should be monotonically ascending (see Fig. 1). If this is not the case, then for each pair $(d_{i,j}, \delta_{i,j})$ two cases, in which changes are necessary, can be distinguished

- The distance of the succeeding pair is lower than the current pair's distance. The current pair's distance must be reduced, and the succeeding pair's distance must be increased.
- The distance of the preceding pair is higher than the current pair's distance. The current pair's distance must be increased, and the succeeding pair's distance must be decreased.

In both cases, the target value (i.e. the disparity) for the two distances, which are not in order, is calculated as the average value of the two distances. If more than two succeeding distances do not fulfill the monotonicity constraint, the average over all these distances is used.

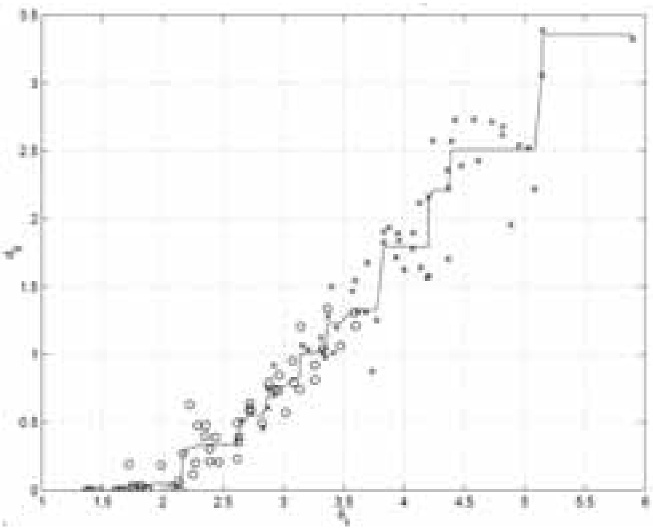


Figure 1: Example scatterplot of distances over dissimilarities (source: [DM00])

With these disparities given, the stress of the current configuration can be calculated and be used to determine the gradient necessary to perform the steepest descent.

2.2.2 Gradient for Steepest Descent

The raw stress over all distances $d_{i,j}$ can be calculated as the sum of the squared deviations of each distance from the according disparity:

$$S' = \sum_{j<i} (d_{i,j} - \hat{d}_{i,j})^2 \tag{4}$$

This value can be normalized with the squared sum of all distances:

$$T' = \sum_{j<i} d_{i,j}^2 \tag{5}$$

such that the normalized and square-rooted stress (see also Eqn. 3) is given by

$$S = \sqrt{\frac{\sum_{j<i} (d_{i,j} - \hat{d}_{i,j})^2}{\sum_{j<i} d_{i,j}^2}} = \sqrt{\frac{S'}{T'}} \tag{6}$$

The gradient is determined using the partial derivatives $\partial d / \partial x_{i,j}$ of the stress function:

$$g_i = \frac{S}{d_{i,j}} \cdot \sum_{(i,j),j<i} \frac{1}{d_{i,j}} \cdot \left[\frac{d_{i,j} - \hat{d}_{i,j}}{S'} - \frac{d_{i,j}}{T'} \right] \cdot \begin{pmatrix} x_{i1} - x_{j1} \\ x_{i1} - x_{j2} \\ \vdots \\ x_{is} - x_{js} \end{pmatrix} \tag{7}$$

This gradient is weighted with a smoothing factor (usually 0.2) and then added to the current position of i in target space. After applying the appropriate gradients to all positions in the current configuration, the resulting stress of the new configuration is determined. When a minimum is reached, the iteration stops and the current configuration is returned as result of the non-metric MDS.

2.3 Parallel and Distributed Approaches

Some approaches exist to execute MDS in a parallel manner. For example, [PD10] proposes a parallel MDS based on a heuristic from particle dynamics. To achieve good parallelisation, the participating nodes (processors) are structured in a stair-like structure as shown in Fig. 2, each node communicating only with few other nodes.

Still, since the arrangement of nodes as well as the collection of results needs to be done by a centralized instance, this approach cannot be applied to Peer-to-Peer systems. This also applies to other parallel implementations as e.g. [CHK$^+$09] and [BBKY06].

To the best knowledge of the authors, the only truly distributed approach for MDS is given in [CPH06]. In this approach, the position of sensors in a sensor network of arbitrary size

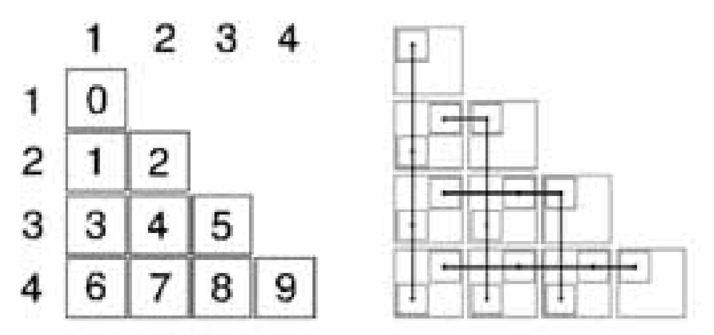


Figure 2: Node arrangement and information flow in [PD10]

is approximated using distance measurements of each single sensor to other sensors in its reach. All other (unreachable) sensors are not considered. The authors show that, if some sensors with fixed (known) positions exist, all other nodes can estimate their one absolute location with good quality. Without having previously fixed positions, still the relative position of each node is found. But, when looking more detailed into the proposed solution, it becomes clear that actually, no real dimensionality reduction is done, because the original data gathered from the sensor can itself already be described in a low-dimension space, even if some additional dimensions may occur due to measurement errors. In their article, the authors assume that their approach could be applied to data of higher dimensions, but provide neither proof nor example for that claim.

3 Algorithm for Distributed Nonmetric MDS

As shown in the previous section, executing an MDS requires global knowledge, i.e. the distances and dissimilarities of all objects need to be known to determine the gradient for each single position. In distributed systems, especially in Peer-to-Peer systems, no single participant has such an omniscient view. It is therefore necessary to run the MDS with only local knowledge.

The basic properties of the proposed approach are:

1. Each peer represents a single object in the high-dimensional space and is able to calculate the dissimilarities to other objects (peers). Since a peer is only a logical entity, a physical peer can still represent several objects.

2. Each (logical) peer is assigned to a 2-D coordinate, representing its position in the current configuration.

3. A peer can only communicate with (and therefore calculate the dissimilarity to) its neighbors. The neighbourhood of each peer is determined using a Voronoi tessella-

tion based on the 2D-coordinates. All peers sharing at least a single Voronoi edge with another peer are set to be a direct neighbour of it. The tessellation is updated locally, whenever a peer's position is changed or a new peer enters the network.

A new Peer P_n is added to the network by successively approximating its position x_P, until an optimum value has been found, at which the peer can be integrated into the network. For the iterative position updates, only the 2D-positions of peers, of whose existence P_n has previously learnt of, are used (sample set). The peer, in whose Voronoi region P_n is currently located, is called the *anchor* peer P_a and may change during iteration. At each position update, P_n gathers the dissimilarity-values and positions of all currently known peers (including the anchor peer as well as P_n itself) and updates its own position such that it is optimal in matters of stress minimization. This can e.g. be done by executing a standard MDS over all known positions, but applying the gradient only to P_n's position and thus leaving all other positions unchanged. If the newly found position lies outside of the Voronoi regions of the current sample set, P_n gets to know additional peers. The positions of these peers are added to the sample set and then taken into account for another update/refinement of x_P. The iteration/refinement stops, when the newly found position lies in the Voronoi-area of any of the already known peers. When this happens, P_n and the current anchor peer reorganize their neighborhood such that P_n is integrated into the network.

The according algorithm goes as follows:

```
Initialization  1.  Contact any random Peer $P_r$
                2.  Set initial ($t = 0$) anchor Peer $P_a^0$ = $P_r$
                3.  Define initial sample set $S^0 = \{P_n, P_a^0\} \cup N(P_a^0)$, where $N(P_a^0)$
                    denotes the set of all direkt neighbors of $P_a^0$.

Iteration $t$   1.  Find position $x_n^{t+1}$ of $P_n$, such that the stress over all positions in
                    $S^t$ is minimized.
                2.  Set new anchor Peer $P_a^{t+1}$ to be the one peer in the (whole) net-
                    work, which has the position with the shortest distance to $x_n^{t+1}$
                3.  If $P_a^{t+1} \notin S^t$
                4.      Add all Peers in direct line between $P_a^t$ and $P_a^{t+1}$ to $S$
                5.      Continue with next iteration step (goto 1.)
                6.  else
                7.      Final position found, reorganize neighbors with $P_a^{t+1}$
                8.      Send position update to all neighbors
```

It is important to point out that finding the anchor peer for any position can easily be done using local algorithms e.g. by means of X-Y routing. This therefore also applies for finding out, which peers are located in direct line between any two positions (see iteration step 4.).

After the new peer has found its final position and has been integrated into the network, it sends a position update into its neighborhood, so that the neighbors can refine their positions by taking P_n's position into account. This position update can be executed in exactly the way as it is done when adding a new peer.

4 Simulation Example

The proposed algorithm has been tested against different dissimilarity matrices both from publicly available questionnaires, and computer generated test-data. The outcome of these simulations is now discussed using the example of a helix[2] over 4.5 periods as high-dimensional data, of which random points have been chosen and been assigned to the peers. After each iteration, the overall stress has been determined.

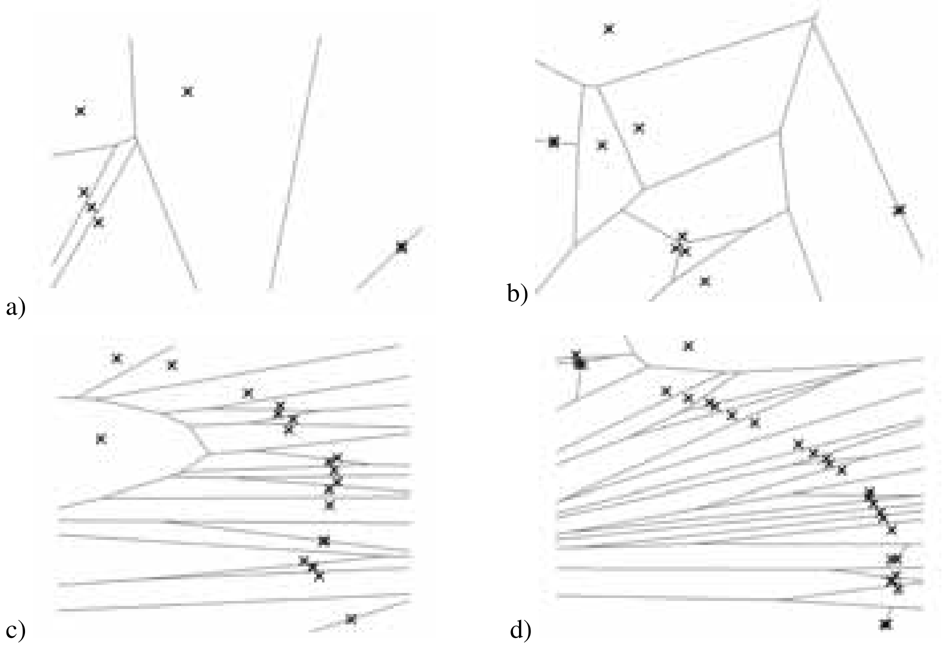


Figure 3: Example network after 7, 11, 22, and 30 peers have been added.

For this example, the proposed algorithm has been run from the first peer on, i.e. no initial global MDS has been executed. Fig. 3a-d) show the configurations after 7, 14, 22, and 30 peers have been added to the network. Each cross represents a peer, while the lines determine the borders of the Voronoi region of each peer. With fewer peers (see Fig3a) and b)), the overall stress assumes values up to $\approx 24\%$, such that the resulting configuration does not allow an intuitive interpretation. Nevertheless, after more peers have been added,

[2] The helix has only been chosen as example, because its projection into 2D is easily understandable, even when a large amount of points are used.

the resulting configurations in c) and d) intuitively seem to be a good 2D-representation of a helix. Indeed, the overall stress of these two configurations is $< 1\%$.

Fig. 4a) shows the development of the stress, when peers are added to the network. It can be seen that when adding only few peers, the stress assumes a large range of values. Nevertheless, with more peers added to the network, good configurations with remarkably low stress values are found.

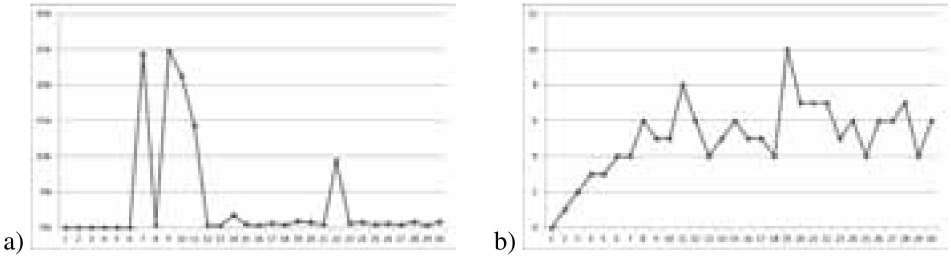
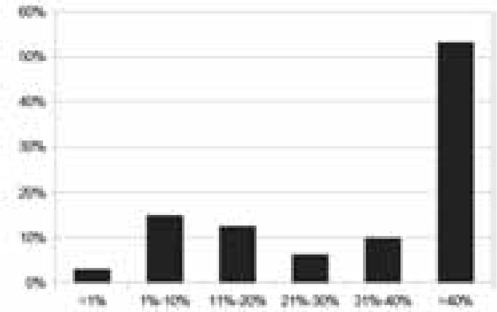


Figure 4: a) stress as a function of number of peers, b) number of peers considered in add operation

Fig. 4b) shows the number of peers, which have been taken into consideration by a newly added peer to find its optimal position in the configuration. It can clearly be seen, that only a small subset (sample) of all peers is necessary to find the position. In few cases (e.g. when adding peer 12 or peer 19), larger portions of the network have to be explored, until the final position is reached. Since for each considered peer, two messages have to be sent to gather that peer's dissimilarity values, the number of considered peers has the same complexity as the number of messages needed to find the optimal position for the new peer. Since the newly added peer can contact several peers at a time to gather the dissimilarity information, the number of considered peers is also an upper bound for the necessary time.

The simulations have also shown that optimal configurations are not always found. Instead, the resulting final stress strongly depends on the initial seed of the pseudorandom number generator that has been used, and therefore both on a) the initial positioning of a newly added peer, and b) on the results of the stress minimization process during each position update. To demonstrate this effect, the algorithm has been executed on the same dataset repeatedly with varying initial seeds for the pseudorandom number generator.

In Fig. 5 it can be seen that only 3% of all simulation runs actually lead to excellent stress values of less than 1%, while another 15% of the simulation runs at least lead to good results with stress values in the range from 1% to 10%. In all other cases, the algorithm gets stuck in local minima when adding a peer to the network, such that a global optimum cannot be reached. This is a well known problem also in globally executed MDS.



Range	Ratio
$< 1\%$	3%
1% to 10%	15%
11% to 20%	13%
21% to 30%	6%
31% to 40%	10%
$> 40\%$	53%

Figure 5: Distribution of final stress when varying the initial random seed

5 Preliminary Conclusions and Future Work

The various simulations run so far indicate that the proposed algorithm is able to solve the problem of distributed non-metric MDS with results of the same quality as a non-metric MDS with global knowledge. Although further investigation of the algorithm is necessary to provide a profound analysis, the following preliminary conclusions can be made:

1. A good starting configuration found by using a global MDS is necessary to achieve good results and reliable behaviour. Otherwise, the algorithm tends to get stuck in local minima when a single peer updates its position. This leads to unsatisfying overall configurations, when new peers are added.
2. Several initial anchor peers should be used to avoid local minima. This is the distributed analogue to using different initial configurations in global MDS.
3. Position changes (updates) should only be propagated to neighbouring peers, if the distance between old and new position exceeds some threshold. This avoids network load and alternating position updates of neighboured peers.
4. The time and message complexity for add operations have been in $O(\sqrt{n})$ for all simulations.

After the preliminary conclusions have been strengthened through further simulations and analysis, future work will focus on finding robust starting configurations that allow to find globally optimal configurations as good as possible. Additionally, an estimation for the number of bootstrapping peers necessary to find satisfying configurations, shall be found.

References

[BBKY06] M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh. Multigrid multidimensional scaling. *Numerical Linear Algebra with Applications*, 13(2-3):149–171, 2006.

[CHK$^+$09] Raimondas Ciegis, David Henty, Bo Kagström, Julius Zilinskas, and Julius Zilinskas. Parallel Global Optimization in Multidimensional Scaling. In *Parallel Scientific Computing and Optimization*, volume 27 of *Springer Optimization and Its Applications*, pages 69–82. Springer New York, 2009.

[CPH06] Jose A. Costa, Neal Patwari, and Alfred O. Hero, III. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Trans. Sen. Netw.*, 2(1):39–64, February 2006.

[CU12] Hauke Coltzau and Bastian Ulke. Navigation in the P2Life Networked Virtual Marketplace Environment. In Herwig Unger, Kyandoghere Kyamaky, and Janusz Kacprzyk, editors, *Autonomous Systems: Developments and Trends*, volume 391 of *Studies in Computational Intelligence*, pages 213–227. Springer Berlin / Heidelberg, 2012.

[dL77] Jan de Leeuw. Applications of convex analysis to multidimensional scaling. *Recent developments in statistics*, page 133145, 1977.

[DM00] T. Denoeux and M. Masson. Multidimensional Scaling of Interval-Valued Dissimilarity Data. *Pattern Recognition Letters*, 21:21–83, 2000.

[Kru64a] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964. 10.1007/BF02289565.

[Kru64b] J. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, June 1964.

[PD10] Piotr Pawliczek and Witold Dzwinel. Parallel Implementation of Multidimensional Scaling Algorithm Based on Particle Dynamics. In Roman Wyrzykowski, Jack Dongarra, Konrad Karczewski, and Jerzy Wasniewski, editors, *Parallel Processing and Applied Mathematics*, volume 6067 of *Lecture Notes in Computer Science*, pages 312–321. Springer Berlin / Heidelberg, 2010.

[Tor52] Warren Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17:401–419, 1952. 10.1007/BF02288916.