Eigensolver Methods for Progressive Multidimensional Scaling of Large Data

Ulrik Brandes and Christian Pich

Department of Computer & Information Science, University of Konstanz, Germany
{Ulrik.Brandes,Christian.Pich}@uni-konstanz.de

Abstract. We present a novel sampling-based approximation technique for classical multidimensional scaling that yields an extremely fast layout algorithm suitable even for very large graphs. It produces layouts that compare favorably with other methods for drawing large graphs, and it is among the fastest methods available. In addition, our approach allows for progressive computation, i.e. a rough approximation of the layout can be produced even faster, and then be refined until satisfaction.

1 Introduction

The term multidimensional scaling (MDS) refers to a family of techniques for dimensionality reduction that are used to represent high-dimensional data in low-dimensional space while approximately preserving distances. For drawing graphs, methods based on the objective function of *distance scaling* are used widely, but the *classical scaling* approach has only occasionally been recognized as a useful alternative [7, 21, 24]. Indeed the computational complexity of this method is quadratic in the input size and thus prohibitive for large graphs.

In this paper we propose a sampling-based approximation technique to overcome this restriction and to reduce time and space complexity essentially to linearity. The proposed algorithm is simple to implement, yet extremely fast and therefore applicable to very large graphs. Moreover, it allows for progressive computation by very quickly producing a rough approximation of the layout, which can then be improved by successive refinement.

This paper is organized as follows. Background on multidimensional scaling and derived methods is provided in Section 2. In Section 3 we introduce two variants of the eigensolver approach, which are evaluated and compared to each other in Section 4. Section 5 concludes our contribution.

2 Related Work

The first MDS algorithm is due to Torgerson [30] and nowadays referred to as *classical MDS* or *classical scaling*. Its objective is a low-dimensional representation of high-dimensional data by fitting inner products; it has a global optimum which can be directly computed by spectral decomposition. The method we propose in this paper is an efficient approximation of classical scaling.

Another MDS variant best known and most widely used today has been proposed by Kruskal [22] and is sometimes distinguished as *distance scaling*. The objective is to directly fit Euclidean distances in the drawing to the given graph-theoretical distances, typically by minimizing a stress measure. It is performed by iterative replacement according to a spring model of attracting and repelling forces or an energy model, as widely known in the graph drawing literature [18], or by iterative algebraic techniques [12]. Due to their time and space complexity, straightforward implementations of distance scaling methods are restricted to data sets of moderate cardinality.

In the graph drawing literature, methods based on linear algebra have become popular in recent years. Examples are High-Dimensional Embedding (HDE) [15], fast multiscale approaches based on eigenvectors of the Laplacian [20], subspace-restricted layout [19], and stress majorization [12].

Poor scalability to large data sets due to quadratic complexity is a well-known problem of all MDS algorithms. It was addressed as early as in the 1960s [23], and since then, many approaches to speeding up spring-force computations have been devised [6, 16, 25, 26]. Likewise, methods for speeding up the spectral methods have been proposed [11, 31]. Closest to our approach is Landmark MDS [10]; we give an experimental comparison in Sect. 4. Relationships between these approaches are discussed in [2, 27]. For more general surveys on sparse techniques for dimensionality reduction and related spectral methods see [5, 28].

MDS seems to have been the first computerized layout method used for drawing social networks [17] towards the end of the 1960s. Even in this restricted application domain there are many extensions and variants, such as incremental or interactive MDS [1, 4, 8, 32]. For further information about MDS, its history, and other applications we refer the reader to recent textbooks [3, 9].

3 Multidimensional Scaling and its Approximation

Let $\Delta \in \mathbb{R}^{n \times n}$ denote a symmetric matrix of metric *dissimilarities* or *distances* δ_{ij} between items $i, j \in \{1, \ldots, n\}$. The goal of multidimensional scaling is to find positions $x_i \in \mathbb{R}^d$ in d-dimensional space, $d \ll n$, such that $\|x_i - x_j\| \approx \delta_{ij}$, i.e. distances are represented well in this low-dimensional space. Note that for notational convenience we write positions x_i as column vectors, and that $d \in \{2, 3\}$ for visualization purposes. With $\Delta^{(2)}$ we denote matrix Δ with squared entries, i.e. $[\Delta^{(2)}]_{ij} = [\Delta]_{ij}^2$.

In graph drawing and network analysis, Δ frequently consists of shortest-path distances (see, e.g., [8] for an alternative graph distance). In other contexts it is often induced by a high-dimensional feature space with an associated distance function.

In this section, we briefly describe a standard technique for multidimensional scaling, a recently introduced method for its fast approximation, and our new variant of this approximation. It turns out that, technically, our method is very similar to one of the fastest algorithms for drawing large graphs [15], but eliminates some of its shortcomings. This is outlined in Sect. 3.5.

3.1 Classical MDS

We briefly describe the scaling method known as Classical MDS [30]. Recall that we are looking for an embedding in d-dimensional space, i.e. a matrix $X \in \mathbb{R}^{n \times k}$ with $X = [x_1, \ldots, x_n]^T$, such that $\delta_{ij} \approx \|x_i - x_j\|$. Since this implies

$$\delta_{ij}^2 \approx \|x_i - x_j\|^2 = (x_i - x_j)^T (x_i - x_j) = x_i^T x_i - 2x_i^T x_j + x_j^T x_j,$$

consider the matrix $B = XX^T$ of inner products $b_{ij} = x_i^T x_j$. While we do not know X, it can be shown that

$$b_{ij} = -\frac{1}{2}\left(\delta_{ij}^2 - \frac{1}{n}\sum_{r=1}^{n}\delta_{rj}^2 - \frac{1}{n}\sum_{s=1}^{n}\delta_{is}^2 + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n}\delta_{rs}^2\right),$$

so that B can also be obtained by double-centering the matrix of squared dissimilarities $\Delta^{(2)}$, i.e. each column and each row of B sums to zero.

Knowing B, positions X are reasonably reconstructed using the eigendecomposition $B = V\Lambda V^T$, where Λ is the diagonal matrix of the eigenvalues of B, and V is the orthonormal matrix of its eigenvectors. Simply let

$$X = V(d)\Lambda_{(d)}^{1/2} \ ,$$

where $\Lambda_{(d)} \in \mathbb{R}^{d \times d}$ is the diagonal matrix of the d largest eigenvalues of B and $V(d) \in \mathbb{R}^{n \times d}$ is an $n \times d$ matrix of associated eigenvectors. Thus, the essence of classical scaling is to fit inner products rather than distances as in distance scaling.

It is important to note that the two or three required eigenvectors can be computed by power iteration, i.e. by repeatedly multiplying a starting vector $x \in \mathbb{R}^n$ with B. The iterate is periodically normalized; further eigenvectors are found by orthogonalization against previously computed eigenvectors. See, e.g., [13] for background on matrix computations.

The running time for drawing an unweighted graph with n vertices and m edges by performing classical MDS on its matrix Δ of shortest-path distances is thus $\mathcal{O}(nm)$ for computing Δ using breadth-first search, $\Theta(n^2)$ for constructing B, and another $\mathcal{O}(n^2)$ per iteration. Running times and also storage requirements are therefore prohibitive for large graphs.

3.2 Landmark MDS

Landmark MDS (LMDS) [10] is a fast method for approximating the results of Classical MDS using a sparsification of the transformed distance matrix. It is based on distinguishing a few items as *landmarks*, and computing the eigendecomposition only on the double-centered matrix of squared distances among those landmarks. Positions of non-landmarks are then determined as linear combinations of landmark positions, i.e. items are placed in the weighted barycenter of all landmarks where the weights are derived from the original distances.

The rationale is that a set of appropriate reference points is sufficient to determine the projection into low-dimensional space. To be representative, the k landmarks, $d < k \ll n$, should be distributed well. Common experience shows that a *MaxMin* strategy, in which the next landmark maximizes the minimum distance to the previous landmarks, yields satisfactory results. Note that this corresponds to a well-known 2-approximation of the k-center problem in facility location. We have tried other simple strategies such as *MaxSum*, random selection, and hybrids, but none proved to be superior consistently. More advanced techniques are proposed in [29].

Time and space complexity of LMDS are significantly smaller than for Classical MDS. Landmark selection and distance computations are carried out in $\mathcal{O}(k \cdot |E|)$ time, each power iteration step requires only $\mathcal{O}(k^2)$ time, and the final positioning is done in $\mathcal{O}(kn)$ time. Since, in general, choosing $k < 100$ yields satisfactory results on most practical instances, LMDS can be regarded a linear-time algorithm. Moreover, it is only necessary to store the $\Theta(kn)$ distances to landmarks.

3.3 Pivot MDS

We now introduce a new variant of sparse MDS which we call *Pivot MDS* (PMDS). It is motivated by a potential shortcoming of the LMDS strategy to position landmarks only with respect to each other: it is possible that the (already available) distance information to non-landmarks can be utilized to improve the quality of the result.

Recall that Classical MDS is based on an eigendecomposition of the double-centered $n \times n$-matrix of squared distances B, and that Landmark MDS is based on the corresponding decomposition of the double-centered $k \times k$-submatrix of squared distances among selected items only. Pivot MDS is based on the double-centered $n \times k$-submatrix C of squared distances from every item to those selected, having entries

$$c_{ij} = -\frac{1}{2}\left(\delta_{ij}^2 - \frac{1}{n}\sum_{r=1}^{n}\delta_{rj}^2 - \frac{1}{k}\sum_{s=1}^{k}\delta_{is}^2 + \frac{1}{nk}\sum_{r=1}^{n}\sum_{s=1}^{k}\delta_{rs}^2\right),$$

where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$, and thus contains all distance information available.

Note that the n-dimensional left singular vectors of $C \in \mathbb{R}^{n \times k}$ are equal to the eigenvectors of $CC^T \in \mathbb{R}^{n \times n}$. If they are computed using power iteration, an iteration consists of two steps: first, positions of pivots are determined using the current positions of all items (multiplication with $C^T \in \mathbb{R}^{k \times n}$), and then all items are positioned relative to the pivots (multiplication with $C \in \mathbb{R}^{n \times k}$).[1]

[1] This interpretation motivates the name "pivot," in contrast to "landmarks" which are first assigned their final location and then used to determine the position of all other items.

An intuitive interpretation is that the eigenvectors of CC^T approximate the eigenvectors of B^2, and thus of B. This follows from the assumption

$$\left[B^2\right]_{ij} = \left[BB^T\right]_{ij} = \sum_{\ell=1}^{n} b_{i\ell} b_{j\ell} \approx \sum_{\ell=1}^{k} c_{i\ell} c_{j\ell} = \left[CC^T\right]_{ij},$$

so matrix entries $[B^2]_{ij}$ and $[CC^T]_{ij}$ represent the same type of transformed distance sums, though in the latter case with a truncated list of intermediaries. If these are sufficiently well distributed, the relative size of entries in CC^T is representative for those in B^2.

At face value the iteration time of PMDS is $\mathcal{O}(kn)$. However, we can rewrite $(CC^T)^i = C(C^TC)^{i-1}C^T$ so that the iteration is performed only on the $k \times k$-matrix C^TC. The initial multiplication with C^T can be omitted (in Sect. 3.4 we will argue, though, that it is sometimes desirable), since the starting vector is arbitrary. The final multiplication with C is similar to the final projection step of LMDS. The algorithm is summarized in Alg. 1.

Except for the additional $\mathcal{O}(kn+k^2n)$ cost of double-centering and computing C^TC, the running time is therefore essentially the same as in LMDS.

Algorithm 1: Pivot MDS

Input: undirected graph $G = (V, E)$, number $k \in \mathbb{N}$ of pivots
Output: coordinates $x, y \in \mathbb{R}^n$

select k pivots from V
for $i \in \{1, \ldots, k\}$ **do**
  i-th column of $\Delta(k) \leftarrow$ BFS(i-th pivot)
$C \leftarrow$ doublecenter $\left(\Delta(k)^{(2)}\right)$
$(v_1, v_2) \leftarrow$ poweriterate(C^TC) // 2 largest eigenvectors
$x \leftarrow Cv_1, y \leftarrow Cv_2$

3.4 Progressive MDS

When using pivot approximation there is a natural trade-off between running time and memory usage; users might have to experiment with various numbers of pivots and different strategies. Instead of iteratively re-executing the algorithm with a larger set of pivots for layout improvement, we propose to use a progressive form of MDS computation that we shall describe in the following.

Let $\Delta(k) \in \mathbb{R}^{n \times k}$ denote a submatrix of the matrix of pairwise distances, and let $x \in \mathbb{R}^n$ be a component in the placement computed by PMDS based on it. To improve approximation quality, $\Delta(k)$ can be extended by a certain number of new pivot columns to $\Delta(k') \in \mathbb{R}^{n \times k'}$ $(k' \geq k)$. Note that all operations for computing the new columns in $\Delta(k')$, double-centering of $\Delta(k')^{(2)}$ to obtain C', and determination of matrix C'^TC' can be implemented to run in $\mathcal{O}\left((k'-k) \cdot |E|\right)$. The new vector $x' \in \mathbb{R}^n$ is computed by replacing C with C' in Algorithm 1.

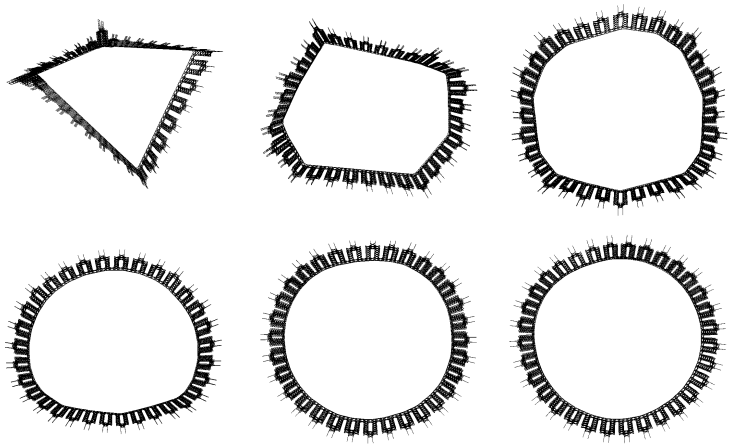

Fig. 1. Progressively drawing the finan512 graph ($|V| = 74752, |E| = 261120$) with increasing pivot set ($k = 3, 6, 12, 25, 50, 100$) using the minmax strategy.

To prevent artificial effects through rotation and reflection in the transition from x to x' due to indeterminacies in the basis of the eigenspace of ${C'}^T C'$, the initial solution $y \in \mathbb{R}^k$ for the power iteration is derived from the previous layout by $y = {C'}^T x$. Compared to random initialization, the iteration process for computing the new layout x' is thus more likely to converge towards a solution close to x, and we have observed that transitions between intermediate layouts tend to become smoother and visually more pleasing.

For smaller graphs, pivots may be added in batches before computing the layout, while it can make more sense for very large graphs to extend $\Delta(k)$ column by column, after each insertion computing the layout anew. In Sect. 4 our experiments indicate that most of the running time of Pivot MDS is consumed by the distance computations, while a layout based on these distances can be computed quickly. It is thus worthwhile to progressively compute the layout until the quality does not improve significantly.

3.5 Pivot MDS vs. HDE

In retrospect, our proposed method is reminiscent of another fast algorithm for drawing large graphs, the high-dimensional embedder (HDE) of [15].

HDE proceeds as follows: From a set of k selected nodes (the pivots), distances to all other nodes are determined. These distances are, however, neither squared nor double-centered, but directly interpreted as coordinates in a k-dimensional space. In this space, they are centered to place the mean at zero coordinate, and yield a high-dimensional embedding $X \in \mathbb{R}^{n \times k}$. This k-dimensional embedding is then projected into two dimensions by Principal Component Analysis (PCA), i.e. by computing the two largest eigenvectors of the

covariance matrix $\frac{1}{n}X^T X \in \mathbb{R}^{k \times k}$. The final coordinates are then obtained by matrix multiplication with X analog to PMDS.

While this appears technically similar to PMDS, it is important to note that both approaches are motivated by different intuitions and produce different results: HDE transforms k possibly correlated variables (the embedding X) into two uncorrelated variables (the layout). In contrast, the objective of PMDS is to directly find low-dimensional coordinates with inner products complying with the given dissimilarities δ_{ij}. More details about the fundamental differences of the two approaches and experiments can be found in [21].

Both PMDS and HDE have approximately the same running time complexity of $\mathcal{O}(k \cdot |E| + k^2 n)$, while PMDS appears to yield drawings of superior quality.

4 Evaluation

The algorithms were implemented in Java SDK 1.4.1. All experiments were conducted under MS Windows XP Version 2002 SP2 on an Intel Pentium-M CPU with 1.6GHz and 512MB of main memory.

We used a set of test graphs for drawing and for evaluating the scalability of our approach. We measured the CPU running times for distance computation and the layout algorithm. Descriptions of the test graphs are given in [14, 15].

4.1 Running Time

Figure 2 shows for both Pivot and Landmark MDS that the running time for the breadth-first searches for matrix C in $\mathcal{O}(km)$ time indeed dominates over the computation times for spectral decomposition of $C^T C$ and the final coordinates, which together are in $\mathcal{O}(k^3 + k^2 n)$ time. The larger the graph and the smaller k in relation to n, the more apparent this effect becomes. LMDS is slightly faster than Pivot MDS because it does not require the construction of $C^T C$.

We have used straightforward, non-optimized implementations for distance computations and matrix operations. Therefore, we expect that using specialized libraries with sophisticated algorithms and data structures yields significant improvements on the absolute values of the measured times.

One important consequence from our observation is that the number of pivots used for the approximation and the pivot strategy can be crucial for the ratio between quality and running time. The next subsection gives more details on the quality of the approximation relative to (full) Classical MDS.

4.2 Quality

To assess their approximation quality, we compared the approximated layouts with those given by full Classical MDS. *Procrustes analysis* (see, e.g., [9]), a technique popular in data analysis and statistics, is used to assess how similar two configurations $X, Y \in \mathbb{R}^{n \times d}$ with $X = [x_1, \ldots, x_n]^T, Y = [y_1, \ldots, y_n]^T$ are up to translation, dilation, and rotation. It is the sum of squared distances

name	$\|V\|$	$\|E\|$	BFS	layout	total
ug380	1104	3231	0.05	0.03	0.08
fidap006	1651	23914	0.16	0.01	0.17
esslingen1	2075	4769	0.07	0.01	0.08
3elt	4720	13722	0.22	0.05	0.27
power	4941	6594	0.17	0.05	0.22
add32	4960	9462	0.15	0.02	0.17
bcsstk33	8738	291583	1.96	0.05	2.01
whitaker3	9800	28989	0.34	0.04	0.38
crack	10240	30380	0.45	0.05	0.50
4elt2	11143	32818	0.41	0.06	0.47
4elt	15606	45878	0.78	0.08	0.86
sphere	16386	49152	0.81	0.09	0.90
fidap011	16614	537374	3.52	0.08	3.60
bcsstk31	35588	572914	4.36	0.19	4.54
bcsstk32	44609	985046	7.09	0.25	7.34
finan512	74752	261120	4.11	0.40	4.50
ocean	143437	409593	10.24	0.82	11.06

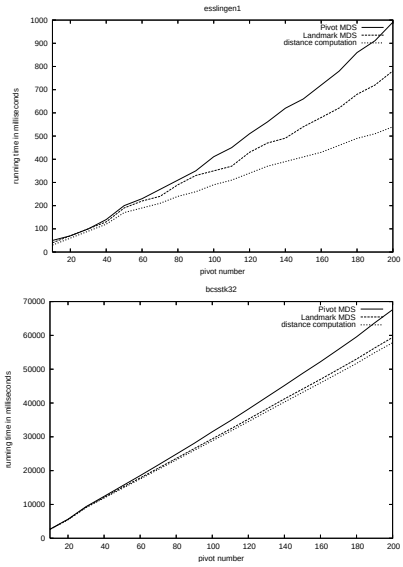

Fig. 2. Left: Pivot MDS running times in seconds using 50 pivots, measured for the set of test graphs. Right: Total running times required by distance computations, Pivot MDS, and Landmark MDS (the latter two including distance computation) with increasing pivot numbers, as measured for ESSLINGEN1 and BCSSTK32.

$$R^2 = \sum_{i=1}^{n} (x_i - y_i)^T (x_i - y_i),$$

where both configurations consist of two-dimensional coordinates (i.e., $d = 2$). Procrustes analysis translates, dilates, and rotates X such that R^2 is minimized with respect to Y. It can be shown (see, e.g., [3]) that $0 \leq R^2 \leq 1$ and that the minimum value is given by the *Procrustes statistic*

$$R^2 = 1 - \frac{\left(\mathrm{tr}(X^T Y Y^T X)^{1/2}\right)^2}{\mathrm{tr}(X^T X)\mathrm{tr}(Y^T Y)},$$

which is the sum the squared distances between X after the best possible transformation (with respect to Y), and Y. If the two configurations can be perfectly matched, $R^2 = 0$; if they cannot be matched at all by any transformation, $R^2 = 1$. We may assume that both configurations have the centroid in the origin.

We computed the R^2 value for the ESSLINGEN1 graph with Pivot and Landmark MDS, using the maxmin and the random pivot strategy, as depicted in Figure 3 with respect to the layout by full MDS. It can be seen that our method is almost consistently superior to Landmark MDS and that it seems to give more stable results. An interesting observation for both algorithms is that using the minmax pivot strategy yields good results with a small number of pivots, while,

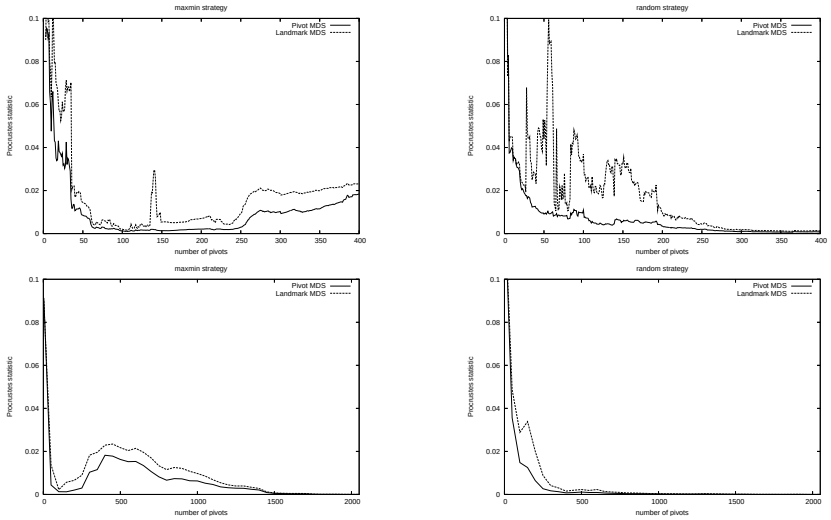

Fig. 3. Procrustes statistic vs. number of pivots for ESSLINGEN1. Upper row: Quality of PMDS and LMDS for practical use ($3 \leq k \leq 400$). Lower row: the same for the full scope ($3 \leq k \leq n$, larger step size in the plot). All curves reach 0 at $k = n$.

starting from a certain point, systematic pivot selection creates an unbalanced approximation leading to deterioration of quality. In contrast, using a random pivot strategy initially requires a larger number of pivots to obtain the same approximation quality, but displays a more monotonic behavior.

As the Procrustes statistic can be computed efficiently, it is suitable for comparing intermediate layouts when increasing the number of pivots, and may be used as a termination criterion. Progression may be stopped when the value of R^2 for consecutive layouts falls below a given threshold, indicating that little to no quality improvement can be expected by adding more pivots.

It is important to note that there are graphs for which Classical MDS (even without approximation) may be of poor quality due to the fact that the two dimensions in the layout are not sufficient for expressing the higher-dimensional structure of the data. In contrast, graphs with a very regular structure, such as finite-element meshes, often have a direct relation between coordinates in a low-dimensional space and graph-theoretical distances, and therefore almost surely yield useful layouts.

This is frequently referred to as the *intrinsic dimensionality* of the data. It can be estimated by the eigenvalue distribution: Few large positive and a large number of "almost zero" (hence rather uninformative) eigenvalues suggest a small number of intrinsic dimensions (which can be captured in a low-dimensional representation well); many large positive eigenvalues indicate a high intrinsic dimensionality and that there is little hope to get a feasible low-dimensional layout with any distance-based method.

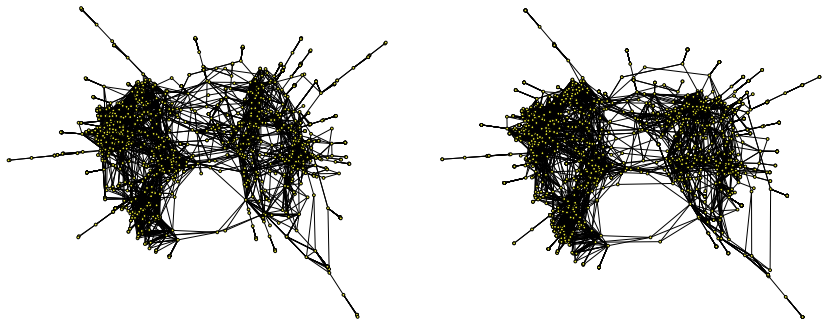

Fig. 4. Layouts of the ESSLINGEN1 graph using Pivot MDS approximation with 50 pivots (left), and by full Classical MDS (or, equivalently, 2075 pivots). The Procrustes statistic yields $R^2 = 0.0085$, indicating an excellent "fit"

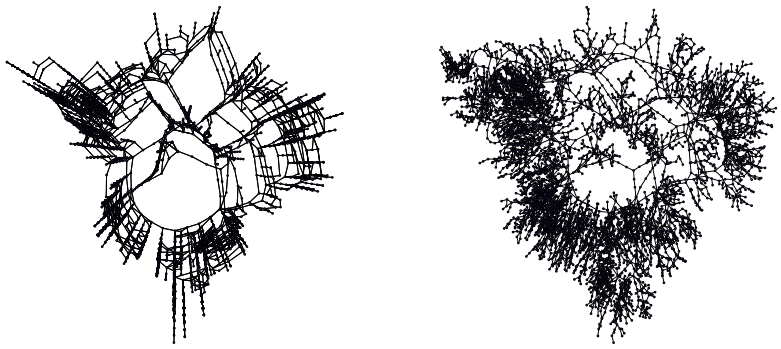

Fig. 5. The US power grid graph ($|V| = 4941, |E| = 6594$). Left: Pivot MDS using 100 pivots. Right: The same after postprocessing by a spring embedder. Pivot MDS appears to give a better layout of the grid structure, while the spring embedder displays regional density better. This suggests the use of our method for efficient generation of initial placements for further processing, which is crucial for many algorithms.

5 Conclusion

We have proposed a simple and efficient method for drawing very large undirected graphs based on MDS. With pivot approximation it can be implemented to run in linear time and with linear memory.

The graph layout can be made progressive by extending the set of pivots incorporated in the layout computation. This allows for quick generation and display of a decent preview layout, which can then be refined by further computation carried out in the background.

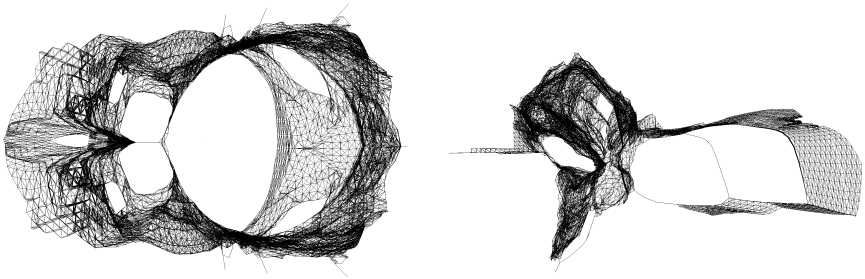

Fig. 6. Drawings of the graphs BCSSTK31 ($|V| = 35588, |E| = 572914$) and BCSSTK32 ($|V| = 44609, |E| = 985046$) with 200 pivots. In the experimental study of [14] these graphs posed serious difficulties for most methods.

In our experiments, we found that generally a very small number of pivots is sufficient and that running time for computing the eigenvectors was negligible with respect to setting up the distance-submatrix C. The essential difference to LMDS is that $C^T C$ contains more relations than just those between landmarks. LMDS and PMDS are therefore equally efficient in practice. We also noted, however, that PMDS indeed requires fewer pivots in general to reach the same quality level, while offering greater overall stability.

Even though our prototypical implementation is written in Java, and we did not perform any optimization, the running times compare favorably with the fastest methods available, and are likely to be reduced significantly in a dedicated implementation.

References

1. W. Basalaj. Incremental multidimensional scaling method for database visualization. In *Proc. VDA*, pages 149–158, 1999.
2. Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In *NIPS*, pages 307–311, 2004.
3. I. Borg and P. Groenen. *Modern Multidimensional Scaling*. Springer, 2005.
4. A. Buja and D. F. Swayne. Visualization methodology for multidimensional scaling. *J. Classification*, 19:7–43, 2002.
5. C. J. C. Burges. Geometric methods for feature extraction and dimensional reduction. Technical report, Microsoft Research, 2004.
6. M. Chalmers. A linear iteration time layout algorithm for visualizing high-dimensional data. In *Proc. InfoVis*, pages 127–132. IEEE, 1996.
7. A. Civril, M. Magdon-Ismail, and E. Bocek-Rivele. SDE: Graph drawing using spectral distance embedding. In *Proc. Graph Drawing*, pages 512–513, 2005.
8. J. D. Cohen. Drawing graphs to convey proximity. *ACM Transactions on Computer-Human Interaction*, 4(3):197–229, 1997.
9. T. Cox and M. Cox. *Multidimensional Scaling*. CRC/Chapman and Hall, 2001.
10. V. de Silva and J. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Proc. NIPS*, pages 721–728, 2003.

11. C. Faloutsos and K. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD*, pages 163–174, 1995.
12. E.R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Proc. Graph Drawing*, pages 239–250, 2004.
13. G. H. Golub and C. F. van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.
14. S. Hachul and M. Jünger. An experimental comparison of fast algorithms for drawing general large graphs. In *Proc. Graph Drawing*, pages 235–250, 2005.
15. D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. In *Proc. Graph Drawing*, pages 388–393, 2002.
16. F. Jourdan and G. Melançon. Multiscale hybrid MDS. In *Proc. IV*, pages 388–393. IEEE, 2004.
17. Charles Kadushin. Personal communication.
18. T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, 1989.
19. Y. Koren. Graph drawing by subspace optimization. In *Proc. VisSym*, pages 65–74, 2004.
20. Y. Koren, L. Carmel, and D. Harel. ACE: A fast multiscale eigenvectors computation for drawing huge graphs. In *Proc. InfoVis*, pages 137–144. IEEE, 2002.
21. Y. Koren and D. Harel. One-dimensional layout optimization, with applications to graph drawing by axis separation. *Computational Geometry: Theory and Applications*, 32:115–138, 2005.
22. J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.
23. J. B. Kruskal and R. E. Hart. A geometric interpretation of diagnostic data from a digital machine: Based on a study of the Morris, Illinois, Electronic Central Office. *Bell Sys. Tech. J.*, 45(8):1299–1338, 1966.
24. J. B. Kruskal and D. Seery. Designing network diagrams. In *Proc. First General Conference on Social Graphics*, pages 22–50, 1980.
25. A. Morrison and M. Chalmers. Improving hybrid MDS with pivot-based searching. In *Proc. InfoVis*, pages 85–90. IEEE, 2003.
26. A. Morrison, G. Ross, and M. Chalmers. A hybrid layout algorithm for subquadratic multidimensional scaling. In *Proc. InfoVis*, pages 152–158. IEEE, 2002.
27. J. C. Platt. FastMap, MetricMap, and Landmark MDS are all Nyström Algorithms. Technical report, Microsoft Research, 2004.
28. L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. Spectral methods for dimensionality reduction. In B. Schölkopf, O. Chapelle, and A. Zien, editors, *Semi-Supervised Learning*. MIT Press, 2006. To appear.
29. J. G. Silva, J. S. Marques, and J. M. Lemos. Selecting landmark points for sparse manifold learning. In *Proc. NIPS*, 2005.
30. W. S. Torgerson. Multidimensional scaling: I. Theory and Method. *Psychometrika*, 17:401–419, 1952.
31. J. T.-L. Wang, X. Wang, K. Lin, D. Shasha, B. A. Shapiro, and K. Zhang. Evaluating a class of distance-mapping algorithms for data mining and clustering. In *Proc. KDD*, pages 307–311, 1999.
32. M. Williams and T. Munzner. Steerable, progressive multidimensional scaling. In *Proc. InfoVis*, pages 57–64. IEEE, 2004.