Multiscale hybrid MDS

Fabien Jourdan, Guy Melançon

Multiscale hybrid MDS

Fabien Jourdan, Guy Melançon
LIRMM, UMR 5506
161 rue Ada
34392 Montpellier Cedex 5, France
fjourdan@lirmm.fr,Guy.Melancon@lirmm.fr

Abstract

We present a multiscale MDS method extending Chalmers' Pivot-based MDS algorithm [14]. Our multiscale strategy is itself based on a $O(N \log N)$ hybrid MDS approach. Our algorithm clearly improves over its predecessors with respect to time, while producing layouts of a comparable quality.

1. Introduction

Multidimensional scaling (MDS) is concerned with the representation of multi-variate data sets as sets of 2D or 3D euclidean points.That is, high dimensional distances or *dissimilarities* between pairs of data elements are used to compute a 2D or 3D view. Although most of the MDS literature focuses on the analysis of the representation and of its *objective* quality [12, 3], efforts have been devoted to the design of methods and algorithms used to actually embed the data in a euclidean space.

Among different approaches, force based placement algorithms have recently gained popularity, partly because of their underlying intuitive model and their ease of implementation. The quality of the drawings they produce can also be accounted for their popularity (see [1, Section 3.5.4]). Although originally designed to produce a drawing of a graph in 2D or 3D space [5, 10, 6], force based placements generalize to abstract datasets, where they can be seen as variations of simulated annealing algorithms. When dealing with a graph, attractive forces correspond to actual links between nodes. Nodes usually are seen as charged masses and repulse each other. A placement algorithm simply runs a simulation of the corresponding physical system, until the overall system reaches a *stable* state which is usually admitted to provide a good view of the data. Algorithms defining variations of the physical model and of the simulation have been published in the past two decades (see [2, 11]).

These approaches can be used when dealing with abstract data where forces are defined according to dissimilarities between data elements, where dissimilarities can be computed from contextual attributes or from any other source. Authors frequently compute dissimilarities as high-dimensional distance between vectors of attributes associated with data elements. As a particular case, nodes of a graph can be embedded in euclidean using this approach, by making the dissimilarity between two nodes coincide with their distance in the graph.

More recent efforts on force-based placements have addressed scalability. Indeed, a straightforward implementation of the simulation requires to iterate a visit of all N^2 pairs of data elements leading to a $O(N^3)$ time complexity for the whole simulation (since most algorithms run the simulation over $O(N)$ iterations). These costly solutions are obviously not relevant as soon as data sets comprise thousands of elements.

Fruchterman and Reingold [7] were among the first to observe that the full visit of the N^2 pairs of elements at each iteration could be avoided, by limiting the computation of the forces to the close neighborhood of nodes, arguing that distant nodes had but a small effect on the overall displacement induced from the repulsive force. Chalmers, Morrison and Ross have fully exploited this avenue in a series of papers on MDS [4, 13, 14]. Their contribution adds to the ideas introduced by Fruchterman and Reingold by restricting the simulation to a subset of randomly chosen elements in a first phase before opening the process to the full data set. Chalmers had already proposed a force-based MDS algorithm running in time $O(N^2)$. The 2002 and 2003 MDS algorithms by Chalmers, Morrison and Ross both used a same idea: run Chalmers' 1996 algorithm on a subset of size $O(\sqrt{N})$ (thus taking $O(N)$ time) and incrementally add all other elements. By astutely selecting the initial positions and active neighborhoods of the added elements, Chalmers *et al.* were able to bring the complexity of their MDS force-based approach as low as $O(N^{5/4})$ [14]. (More detailed comments on the work by Chalmers and his collaborators are provided in the forthcoming sections.)

The algorithms presented here are in continuity with the improvements brought by Chalmers, Morrison and Ross. First, we describe a hybrid MDS approach inspired from their work running in $O(N\log(N))$ and producing output of comparable quality. However, the major contribution of our work is a multiscale hybrid MDS algorithm. Indeed, the sampling approach introduced by Chalmers *et al.* naturally leads to a multiscale schema: instead of applying Chalmers' 1996 algorithm on the subset of size $O(\sqrt{N})$, we recursively apply our $O(N\log N)$ algorithm. As we will show, this approach reveals to be significantly faster and provides output of a quality that compares well with previous approaches.

2. Data sampling, dissimilarities and parent-finding strategies

The 1996 paper by Chalmers [4] had already decreased the complexity of traditional MDS force-based algorithms by designing a linear time iteration approach. This first improvement compared to Fruchterman and Reingold [7], in that at each iteration the placement of a data element depended on a constant number of close neighbors, thus making the overall algorithm $O(N^2)$. The 2002 and 2003 papers by Chalmers, Morrison and Ross [13, 14] introduced a new idea. In a first phase, the computation of the placement is limited to a subset S containing $O(\sqrt{N})$ randomly selected data elements. Applying Chalmers' 1996 quadratic MDS algorithm to the sampled elements thus induced a cost of $O(N)$ for this first phase. The remaining elements were then added iteratively, their position being refined through a series of neighborhood restricted force-based simulations.

Algorithm 1 *Chalmers* et al. *2002 and 2003 algorithms main steps :*

```
Data :    A set of data elements $V$ (of size $N$)
          A dissimilarity measure $\delta(u,v)$
          for all pairs $u,v \in V$
Result :  Euclidean coordinates $(x_u,y_u)$
          for each element $u \in V$

Randomly select a subset $S \subset V$ of size $O(\sqrt{N})$
Embed $S$ in euclidean space using Chalmers 1996
linear time per iteration MDS algorithm.
For each $u \in V \setminus S$ do :
    Find $v \in S$ such that $\delta(u,v)$ is minimum.
    Place $u$ according to $v$'s location.
```

The quality of the placement is partly insured by adequately choosing where to initially position elements u not in S ($u \in V \setminus S$). The 2002 strategy of Chalmers *et al.* was to search for the candidate element in S closest to u (with respect to the dissimilarity measures), introducing a $O(N\sqrt{(N)})$ phase dominating the overall complexity. Their 2003 paper modified this parent-finding strategy. Instead of going through the whole set S they first organized it into smaller components to improve the search. The set S was itself sampled to extract a constant size subset of pivots, all elements of S being assigned to buckets associated with pivots. Each pivot p had thus a list of associated buckets, $b_p^{(1)}, b_p^{(2)}, \dots, b_p^{(k)}$, naturally ordered according to the dissimilarity measures $\delta(-,p)$ or high-dimensional distances from p.

The search for the elements in S being closest to $u \in V \setminus S$ could then be performed by selecting for each pivot p, a candidate element u_p that competed against all others to act as the closest element to u. A search through all emerging candidates u_p, astutely using the bucket structure, allowed to elect the element declared as *closest* to u.

Remark. Others have suggested improvements of force-directed placements based on similar approaches. Kobourov *et al.* [8], for instance, divide the original data set into layers (subsets) that are iteratively taken into account when embedding elements in euclidean space. In another paper [9], Kobourov *et al.* also promote *intelligent placement* of nodes when they are initially introduced in the simulation, as opposed to a simple random initial placement, to improve the overall behaviour of the algorithm.

2.1. Further improving the parent-finding strategy

It is the parent-finding strategy we suggest to change in order to reach a $O(N\log N)$ complexity. Instead of organizing the set S into a subset of pivots with associated buckets, we suggest to associate with each pivot p the list L_p of all sampled elements ordered according to the dissimilarity $\delta(-,p)$ or high-dimensional distance from p. The cost of sorting the list associated with a pivot element p is $O(\sqrt{N}\log(N))$ on average and $O(N)$ in the worst case. The number of pivots being constant, the worst case complexity for this preliminary step is in $O(N)$. Note that, as Chalmers *et al.* admit, their strategy 2003 has a worst case complexity of $O(N\sqrt{(N)})$ thus falling back onto their original 2002 algorithm, which happens in the rare case where all data elements merge into a single bucket.

Using ordered lists as we suggest, the search for the closest candidate to u can be performed by going through each ordered list L_p to find a best candidate u_p, which is done in $O(\log N)$ time. At this point, our strategy differs from that of Chalmers *et al.* Indeed, the candidate u_p is selected by comparing its dissimilarity to p, $\delta(u_p,p)$ to the dissimilarity $\delta(u,p)$. That is, we select the element u_p which is as dissimilar to p as u is. Chalmers *et al.* use a different criterion and select the candidate u_p having a smallest dissimilarity $\delta(u, u_p)$.

Thus, our algorithm can be summarized as follows.

Algorithm 2 *Main steps of our $O(N \log N)$ algorithm:*

```
Data :    A set of data elements $V$ (of size $N$)
          A dissimilarity measure $\delta(u,v)$
          for all pairs $u, v \in V$
Result :  Euclidean coordinates $(x_u, y_u)$
          for each element $u \in V$

Randomly select a subset $S \subset V$ of size $O(\sqrt{N})$
Embed $S$ in euclidean space using Chalmers 1996 algorithm
Select $k$ pivot elements $p$ in $S$ (put them in $P$).
For each element $p \in P$ do :
    Sort all $s \in S$ in a list $L_p$ according to $\delta(s,p)$
For each $u \in V \setminus S$ do :
    Find $u_p$ in $L_p$ such that $|\delta(u,p) - \delta(u,u_p)|$
    is minimum.
    Place $u$ according to $u_p$'s location.
```

The two strategies actually bear some similarity, as we now explain. Assume for a moment that the dissimilarity map δ satisfies the triangular inequality. Assume also that $d(u_p, p)$ compares well with $\delta(u_p, p)$. Then having $\delta(u_p, p)$ close to $\delta(u, p)$ is equivalent to $\delta(u, u_p)$ being small, as illustrated in Figure 1. The assumptions we make actually are reasonable as soon as there are reasons to expect a *good* placement of the data set in the euclidean space. Experimental results presented in the next section confirm that these two strategies produce output of comparable quality.

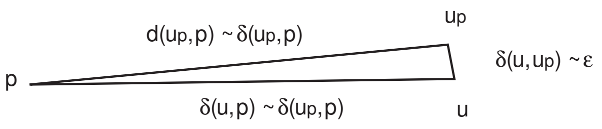

Figure 1. Selecting u_p such that $\delta(u,p)$ approaches $\delta(u_p, p)$ is similar to selecting u_p such that $\delta(u, u_p) \sim \epsilon$.

3. Comparison

As expected, our $O(N \log N)$ strategy improves on Chalmers *et al.* Figure 2 compares the two theoretical curves and exhibits a clear advantage for the $O(N \log N)$ algorithm as N increases. A closer examination of the two curves on a smaller scale indicates that the curves actually meet when $N \sim 5500$, temporarily giving the lead to $N^{5/4}$.

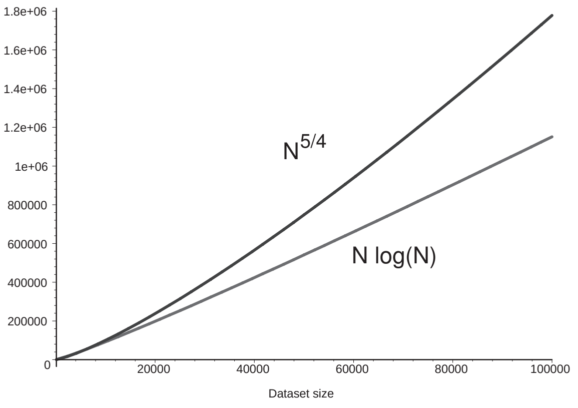

Figure 2. Comparison of the $N^{5/4}$ and $N \log N$ curves on the scale $10^3 - 10^6$.

Clearly, an experimental study is mandatory in order to assess of these theoretical predictions. Indeed, a comparison of the actual running time of the algorithm showed that Chalmers *et al.* 2003 hybrid MDS and our $O(N \log N)$ behave similarly. In other words, the real benefits of a $O(N \log N)$ parent-finding strategy only appear for extremely large N. The actual running time of the three algorithms are reported in Figure 9 (section 4). (They correspond to the three intertwined curves at the top of the figure. The curve drawn in the lower part reports the running time of the multiscale variant described in the next section.)

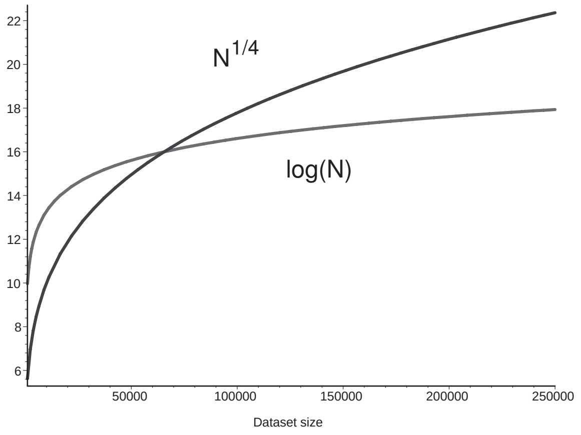

Figure 3. Comparison of the $N^{1/4}$ and $\log N$ curves on the scale $10^3 - 10^6$.

This can be explained. The two algorithms we consider here (Chalmers' et al. 2003 pivot-based MDS and our's) mainly differ on the parent-finding phase. All the other steps of the algorithms behave similarly. Hence, what we ought to compare is the time spent by each algorithm on the parent-finding phase. Incidentally, Chalmers *et al.* made a similar remark and observed that roughly 40% of the time is devoted to the parent-finding process. Now, Chalmers *et al.* parent-finding strategy is in $O(N^{1/4})$ while ours is in $O(\log N)$. Consequently, looking at the curves $N^{1/4}$ and $\log N$ is more relevant when comparing the two algorithms, since these curves predict the time the algorithms respectively spend on the parent-finding process. As Figure 3 shows, the $\log N$ strategy ultimately provides better performance. However, the benefits of the $\log N$ search strategy reveal themselves only for rather large datasets ($\sim 7 \times 10^5$).

This prediction is confirmed by our experiments. Figure 4 reports the actual time spent on the parent-finding process of the $O(N^{5/4})$ and $O(N \log N)$ algorithms. As one can see, the curves grow similarly but start to separate when the datasets reach the predicted size.

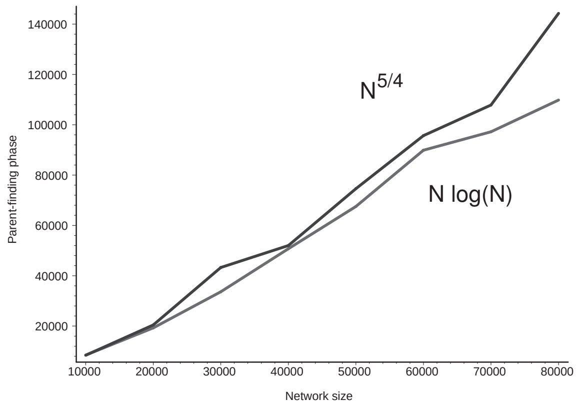

Figure 4. Comparing the actual time spent on the parent-finding phase.

As we will see in a later section, the improvement in time becomes clear and acts as well for smaller dataset when applied through a multiscale schema.

3.1. Quality of the output

The improvement offered by the $O(N \log N)$ algorithm requires that we can assess of the quality of its output. Indeed, improving the running time of force-based MDS only makes sense if we remain able to produce *nice* views of the data. The MDS theory relies on an objective measure to assess of the quality of a placement. This measure is traditionally called the *Stress* of a configuration (or placement) and is defined as:

$$\sigma(X) = \frac{\sum_{u,v} (\delta_{u,v} - d_{u,v})^2}{\sum_{u,v} d_{u,v}^2} \tag{1}$$

where $X : V \to \mathbf{R}^2$ denotes the map defining the embedding of the dataset in euclidean space and where the sums run over all pairs of distinct data elements u, v. Intuitively, the lower the stress of a placement X is, the better the algorithm outputting X was able to satisfy the constraints given by the dissimilarities $\delta_{u,v}$.

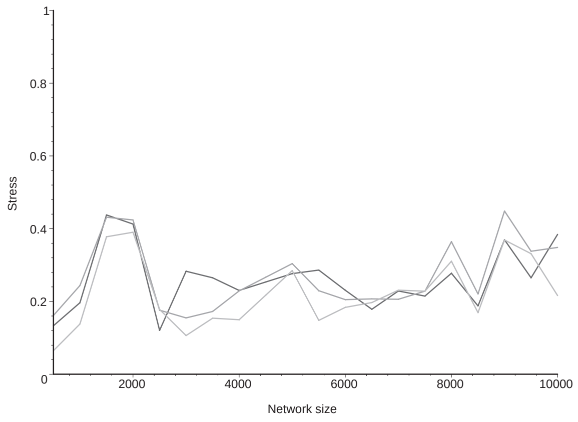

Figure 5. Comparison of Stress for the $O(N^{3/2})$, $O(N^{5/4})$ (grayed) and the $O(N \log N)$ (darker) MDS strategies.

Our experiment showed that our $O(N \log N)$ algorithm behaves similarly to the 2002 and 2003 algorithms by Chalmers *et al.*, as shown in Figure 5. We used as a benchmark a set of randomly generated small world networks of sizes ranging from 500 to 80000 nodes. The algorithm used to generate those graphs randomly selects points in a 2D euclidean space according to a multiple gaussian distribution. More precisely, the distribution used is such that it selects groups of closely situated points[1]. What should be observed is the organization of the points in loosely connected clusters.

Thus, each point in the original data comes equipped with its own 2D coordinates, which moreover determines how it links with its close neighbours and optionally with more distant points. Thus, a MDS algorithm can be fed with dissimilarities exactly equal to the 2D euclidean distance between the points. This type of data clearly acts as a

1 See Figures 6 and 7. The size and resolution of the images are clearly inadequate. Moreover, giving an example of a 10000 nodes network is useless here. Other images can be found at the URL `http://www.lirmm.fr/~fjourdan`).

benchmark for any MDS algorithm which should be able to recover the initial positions of the points, up to obvious symmetries (local combinations of rotations and reflections). Figure 7 illustrates the result of our $O(N \log N)$ algorithm on the 500 elements network of Figure 6. Each of the clusters in the original map can be easily identified by visual inspection[2].

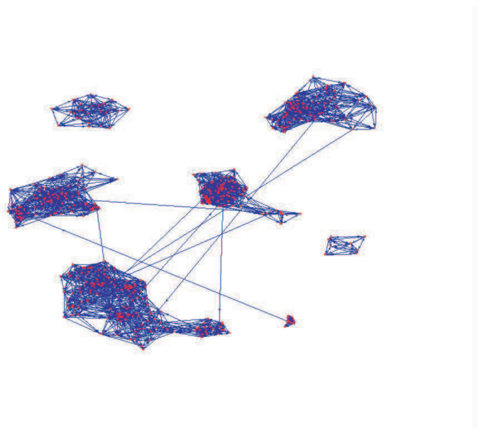

Figure 6. Small world network induced from randomly selected points in 2D (500 node elements).

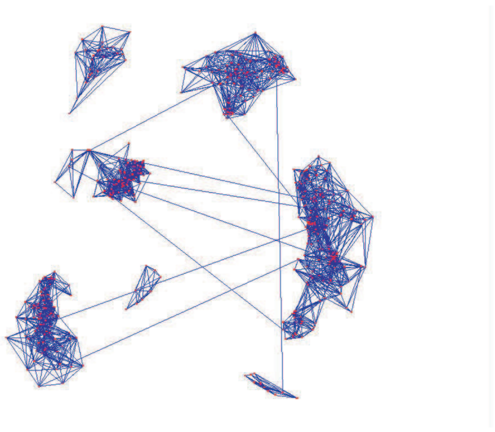

Figure 7. MDS output obtained from the network in Figure 6.

4. Multiscale MDS

Chalmers *et al.* algorithms and the $O(N \log N)$ algorithm described above are two-steps algorithms. They concentrate on a subset of randomly sampled elements, embed it in euclidean space, before developing different strategies to agglomerate the remaining data elements around this kernel configuration. As the kernel configuration is computed using Chalmers' 1996 algorithm, and because this algorithm is in $O(N^2)$, the size of the sampled subset must necessarily be of size $O(\sqrt{N})$ (thus insuring a $O(N)$ cost for this first phase).

In our view, these choices follow a basic assumption : a simulation taking into account a larger range of dissimilarity values produces output of a better quality. Hence, that Chalmers' 1996 algorithm performs better on that aspect than the 2002 and 2003 variants, or the one presented in the previous section, for instance. This assumption is obviously hard to verify and, in our opinion, is incorrect. Depending on the original dataset, it might well be better to first limit the number of dissimilarities defining the forces acting in the simulation, and incrementally expand the simulation to a larger set of dissimilarities. Intuitively, iteratively displacing elements among a crowd may be counterproductive whether incrementally assigning and updating positions may produce good quality output, as our work and that of Chalmers *et al.* assess. Indeed, Chalmers *et al.* had already observed that their $O(N^{3/2})$ strategy gave output showing a lower stress than that of the 1996 algorithm. Figure 7 corroborates our claim and shows the stress values reached by all four algorithms. Clearly, the Stress of our multiscale MDS is lower than that of all three other algorithms (darker curve at the bottom).

Hence, there are evidence that a multiscale (hybrid) force-based MDS can produce output of a quality at least equal if not better than any other force-based MDS. The second benefit we observed from the multiscale schema is a significant improvement in time. Recursively calling the multiscale MDS strategy to compute the kernel configuration clearly improves the performance of the algorithm. As illustrated Figure 9, the improvement brought by the multiscale schema increases with the size of the dataset. Our experiments indicate a ratio stabilizing around 0.4 in favor of the multiscale schema (thus being 60% faster than the two-phase approaches).

Conclusion and future work

The $O(N \log N)$ hybrid MDS algorithm we presented turns out to be faster than Chalmers *et al.* 2002 and 2003 variants for large datasets. The improvement we suggest

2 The experiments were run using the GVF Java API (see `http://gvf.sourceforge.net`). Using the framework we were able to test the whole algorithm and compute the Stress of configurations for dataset of size up to 10000. The framework allowed us to run the parent-finding phase of the algorithms with datasets having a size up to 80000.

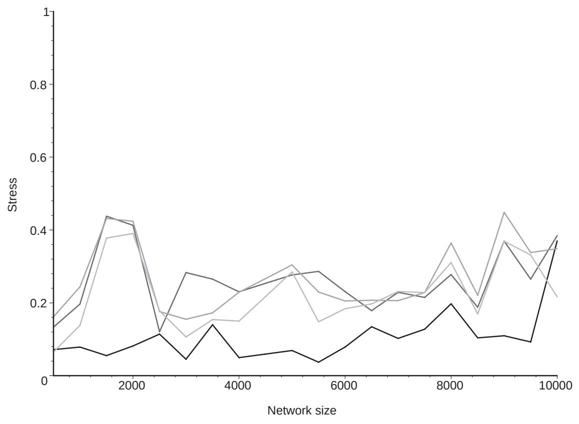

Figure 8. Comparison of Stress for all four MDS algorithms considered here. The darker curve at the bottom reports Stress values reached by the multiscale MDS.

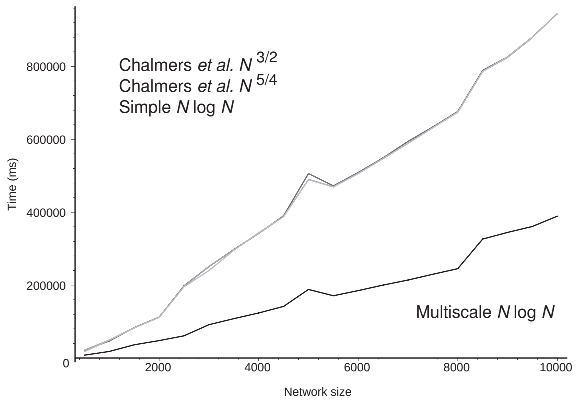

Figure 9. Comparison of the actual running time for all four MDS algorithms considered here.

mainly concerns the parent-finding process involved in the second phase of those hybrid MDS approaches, while all other subroutines show similar performance.

However, when embedding data through a multiscale schema, the improvement in time of the overall algorithm becomes clearer. Furthermore, our experiments enabled us to verify that the output produced both from the simple two-phase $O(N \log N)$ and the multiscale approach are of a quality similar to Chalmers *et al.* 2002 and 2003 hybrid MDS algorithms.

The multiscale approach should be further examined. Indeed, at the top level, the multiscale schema is recursively applied to embed the $O(\sqrt(N))$ sampled elements into a kernel configuration. The placement of the remaining elements is then accomplished just as in the simple $O(N \log N)$ algorithm. Parameters such as the size of the kernel configuration should also be studied in order to ”measure” their impact on the overall quality of the layout. Finally, a comparative study of Stress, normalized Stress and Energy [1, Section 3.1] could eventually lead to finer conclusions about the relative behavior and *objective* quality of all those hybrid, simple and multiscale MDS approaches.

Acknowledgments

We wish to thank Nicolas Colliot for providing the small world networks used for benchmarking the algorithms.

References

[1] W. Basalaj. *Proximity Visualization of Abstract Data*. Ph. D., University of Cambridge, 2000.

[2] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[3] I. Borg and P. Groenen. *Modern multidimensional scaling*. Springer-Verlag, New York, 1997.

[4] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *IEEE Symposium on Information Visualization*, pages 127–132, 1996.

[5] Eades. A heuristic for graph drawing. In *Congressus Numerantium*, volume 42, pages 149–160, 1984.

[6] Frick, Ludwig, and Mehldau. A fast adaptive layout algorithm for undirected graphs. In *Lecture Notes in Computer Science*, volume 894, pages 388–403, 94.

[7] T. M. J. Fruchterman and E. M. Reingold. Graph Drawing by Force-directed Placement. In *Software-Practice and Experience*, volume 21(11), pages 1129–1164. Nov. 1991.

[8] P. Gajer, M. T. Goodrich, and S. G. Kobourov. A Fast Multi-Dimensional Algorithm for Drawing Large Graphs. In *Graph Drawing 2000*, 2000.

[9] P. Gajer, M. T. Goodrich, and S. G. Kobourov. GRIP : Graph dRawing with Intelligent Placement. In *Graph Drawing 2000*, 2000.

[10] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. In *Information Processing Letters*, volume 31, pages 7–15, Apr. 1989.

[11] M. Kaufmann and D. Wagner. *Drawing Graphs*. Springer, 2001.

[12] J. B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, 1978.

[13] A. Morrison, G. Ross, and M. Chalmers. A hybrid layout algorithm for sub-quadratic multidimensional scaling. In *IEEE Symposium on Information Visualization*, pages 152–, 2002.

[14] A. Morrison, G. Ross, and M. Chalmers. Fast multidimensional scaling through sampling, springs and interpolation. In *IEEE Symposium on Information Visualization*, pages 68–77, 2003.