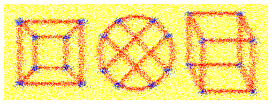

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 21, no. 5, pp. 791–821 (2017)
DOI: 10.7155/jgaa.00440

A Sparse Stress Model

Mark Ortmann Mirza Klimenta Ulrik Brandes

Department of Computer & Information Science
University of Konstanz, Konstanz, Germany

Abstract

Force-directed layout methods are among the most common approaches for drawing general graphs. Among them, stress minimization produces layouts of comparatively high quality while also imposing comparatively high computational demands. We propose a speed-up method based on the aggregation of terms in the objective function. It is akin to aggregate repulsion from far-away nodes during spring embedding but transfers the idea from the layout space into a preprocessing phase. An initial experimental study informs a method to select representatives, and subsequent more extensive experiments indicate that our method yields better approximations of minimum-stress layouts in less time than related methods.

Submitted:	Reviewed:	Revised:	Reviewed:	Revised:
December 2016	March 2017	April 2017	June 2017	June 2017
	Accepted:	Final:	Published:	
	July 2017	July 2017	October 2017	
	Article type:		Communicated by:	
	Regular paper		Y. Hu and M. Nöllenburg	

We gratefully acknowledge financial support from Deutsche Forschungsgemeinschaft under grant Br 2158/11-1

A preliminary version of this paper appeared in the proceedings of 24th Intl. Symp. Graph Drawing (GD'16) [31]

E-mail addresses: Mark.Ortmann@uni-konstanz.de (Mark Ortmann) mail@mirzaklimenta.com (Mirza Klimenta) Ulrik.Brandes@uni-konstanz.de (Ulrik Brandes)

1 Introduction

There are two main variants of force-directed layout methods, expressed either in terms of forces to balance or an energy function to minimize [3, 27]. For convenience, we refer to the former as spring embedders and to the latter as multidimensional scaling (MDS) methods.

Force-directed layout methods are in wide-spread use and of high practical significance, but their scalability is a recurring issue. Besides investigations into adaptation, robustness, and flexibility, much research has therefore been devoted to speed-up methods [22]. These efforts address e.g., the speed of convergence [12, 13] or the time per iteration [1, 19]. Generally speaking, the most scalable methods are based on multi-level techniques [15, 20, 23, 36].

Experiments [5] suggest that minimization of the stress function [29]

$$s(X) = \sum_{i<j} w_{ij}(||X_i - X_j|| - d_{ij})^2 \tag{1}$$

is the primary candidate for high-quality force-directed layouts $X \in \mathbb{R}^{n\times 2}$ of a simple undirected graph $G = (V, E)$ with $V = \{1, \ldots, n\}$, $m = |E|$, and X_i being the two dimensional position of $i \in V$ in X. The target distances d_{ij} are usually chosen to be the graph-theoretic distances, the weights set to $w_{ij} = 1/d_{ij}^2$, and the dominant method for minimization is majorization [18]. Several variant methods reduce the cost of evaluating the stress function by involving only a subset of node pairs over the course of the algorithm [6, 7, 15]. If long distances are well represented already, for instance because of initialization with a fast companion algorithm, it has been suggested that one restricts further attention to short-range influences from l-neighborhoods only [5].

Here we propose to stabilize the sparse stress function restricted to 1-neighborhoods [5] with aggregated long-range influences inspired by the use of Barnes & Hut approximation [1] in spring embedders [34]. Extensive experiments suggest how to determine representatives for individually weak influences, and that the resulting method represents a favorable compromise between efficiency and quality.

Related work is discussed in more detail in the next section. Our approach is derived in Section 3, and evaluated in Section 4. We conclude in Section 5.

2 Related Work

While we are interested in approximating the full stress model of Eq. (1), there are other approaches capable of dealing with given target distances such as the strain model [4, 10, 26] or the Laplacian [21, 28].

An early attempt to make the full stress model scale to large graphs is GRIP [15]. Via a greedy maximal independent node set filtration, this multi-level approach constructs a hierarchy of more and more coarse graphs. While a sparse stress model calculates the layout of the coarsened levels, the finest level is drawn by a localized spring-embedder [13]. Given the coarsening hierarchy

for graphs of bounded degree, GRIP requires $\mathcal{O}(nq^2)$ time and $\mathcal{O}(nq)$ space with $q = \log \max\{d_{ij} : i, j \in V\}$.

Another notable attempt has been made by Gansner et al. [17]. Like the spring embedder, the maxent-model is split into two terms:

$$\sum_{\{i,j\}\in E} w_{ij}(||X_i - X_j|| - d_{ij})^2 - \alpha \sum_{\{i,j\}\notin E} \log ||X_i - X_j||$$

The first part is the 1-stress model [4, 15], while the second term tries to maximize the entropy. Applying Barnes & Hut approximation technique [1], the running time of the maxent-model can be reduced from $\mathcal{O}(n^2)$ per iteration to $\mathcal{O}(m + n \log n)$, e.g., using quad-trees [32, 35]. In order to make the maxent-model even more scalable, Meyerhenke et al. [30] embed it into a multi-level framework, where the coarsening hierarchy is constructed using an adapted size-constrained label propagation algorithm.

Gansner et al. [16], inspired by the idea of decomposing the stress model into two parts, proposed COAST. The main difference between COAST and maxent is that it adds a square to the two terms in the 1-stress part and that the second term is quadratic instead of logarithmic. Transforming the energy system of COAST allows to apply fast-convex optimization techniques making its running time comparable to the maxent model.

While all these approaches somewhat steer away from the stress model, MARS [25] tries to approximate the solution of the full stress model. Building on a result of Drineas et al. [11], MARS requires only $t \ll n$ instead of n single-source shortest path computations. Reconstructing the distance matrix from two smaller matrices and by setting $w_{ij} = 1/d_{ij}$, MARS runs in $\mathcal{O}(tn + n \log n + m)$ per iteration with a preprocessing time in $\mathcal{O}(t^3 + t(m + n \log n) + t^2 n)$, and a space requirement in $\mathcal{O}(nt)$.

3 Sparse Stress Model

The full stress model, Eq. (1), is in our opinion the best choice to draw general graphs, not least because of its very natural definition. However, its $\mathcal{O}(n^2)$ running time per iteration and space requirement, and expensive processing time of $\mathcal{O}(n(m + n \log n))$, hamper its way into practice.

The reason sparse stress models are still in early stages of development is that their application to large graphs requires not just a reduction in the running time per iteration, but also the preprocessing time and its associated space requirement. Where these problems originate from is best explained by rewriting Eq. (1) to the following form:

$$s(X) = \sum_{\{i,j\}\in E} w_{ij}(||X_i - X_j|| - d_{ij})^2 + \sum_{\{i,j\}\in \binom{V}{2}\setminus E} w_{ij}(||X_i - X_j|| - d_{ij})^2 \quad (2)$$

As minimizing the first term only requires $\mathcal{O}(m)$ computations and all d_{ij} are part of the input, solving this part of the stress model can be done effi-

ciently. However, the second term requires an all-pairs shortest path computation (APSP), $\mathcal{O}(n^2)$ time per iteration, and in order to stay within this bound $\mathcal{O}(n^2)$ additional space. We note that the 1-stress approaches presented in Section 2 of Gajer et al. [15] and Brandes & Pich [4] ignore the second term, while Gansner et al. [16, 17] replace it. Discounting the problems arising from the APSP computation, we can see that the spring embedder suffered from exactly the same problem, namely the computation of the second term – there called repulsive forces. Barnes & Hut introduced a simple, yet ingenious and efficient solution, namely to approximate the second term by using only a subset of its addends.

To approximate the repulsive forces operating on node i, Barnes & Hut partition the graph. Associated with each of these $\mathcal{O}(\log n)$ partitions is an artificial representative, a so called super-node, used to approximate the repulsive forces of the nodes in its partition affecting i. However, as these super-nodes only have positions in the Euclidean space, but no graph-theoretic distance to any node in the graph, they cannot be processed in the stress model. Furthermore, deriving a distance for a super-node based on the graph-theoretic distances of all the nodes it represents appears to be both too costly and a poor approximation since the partitioning is computed in the layout space. Choosing a node from the partition as a super-node would not solve the problems, not least because the partitioning changes over time.

Therefore, adapting this approach cannot be done in a straightforward manner. However, the model we are proposing sticks to its main ideas. In order to reduce the complexity of the second term in Eq. (2), we restrict the stress computation of each $i \in V$ to a subset $\mathcal{P} \subseteq V$ of $k = |\mathcal{P}|$ representatives, from now on called *pivots*. The resulting sparse stress model, where $N(i)$ are the neighbors of i and w'_{ip} are adapted weights, has the following form:

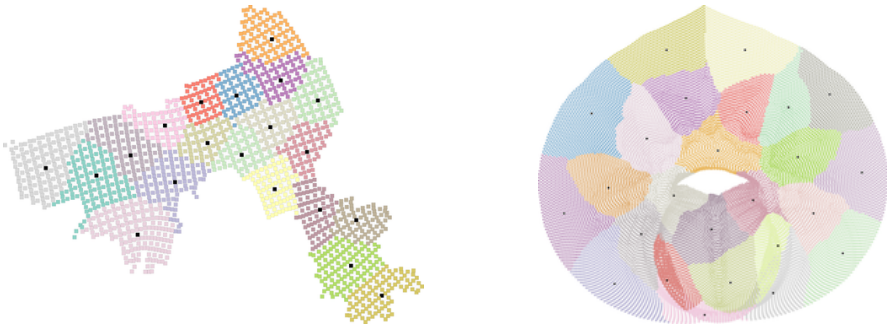
$$s'(X) = \sum_{\{i,j\}\in E} w_{ij}(||X_i - X_j|| - d_{ij})^2 + \sum_{i\in V} \sum_{p\in\mathcal{P}\setminus N(i)} w'_{ip}(||X_i - X_p|| - d_{ip})^2 \quad (3)$$

Note that the Glint framework [24] uses a similar function. However, in contrast to our proposal, it does not involve the first term and the set of pivots in the second term differs for each node $i \in V$. Consequently, this approach requires in the worst-case an APSP computation and therefore is not a sparse stress model in the narrow sense of the definition.

Just like Barnes & Hut, we associate with each pivot $p \in \mathcal{P}$ a set of nodes $\mathcal{R}(p) \subseteq V$, where $p \in \mathcal{R}(p)$, $\bigcup_{p\in\mathcal{P}} \mathcal{R}(p) = V$, and $\mathcal{R}(p) \cap \mathcal{R}(p') = \emptyset$ for $p, p' \in \mathcal{P}$. However, we propose to use only one global partitioning of the graph that does not change over time. Still, just like the super-nodes, we want that the pivots are representative for their associated region. In terms of the localized stress minimization algorithm [18], this means that we want for each $i \in V$ and $p \in \mathcal{P}$

$$\frac{\sum_{j\in\mathcal{R}(p)\setminus N(i)} w_{ij}(X_j^\alpha + d_{ij}(X_i^\alpha - X_j^\alpha)/||X_i - X_j||)}{\sum_{j\in\mathcal{R}(p)} w_{ij}} \approx X_p^\alpha + \frac{d_{ip}(X_i^\alpha - X_p^\alpha)}{||X_i - X_p||},$$

where X^α denotes a single dimension of the layout. As the left part is the



$k = 20$ $k = 25$

Figure 1: Nodes to pivot assignment computed via Alg. 2 for (left) *plat1919* and (right) *bodyy5*. Pivots are colored black and nodes belonging to the same pivot are encoded in the same color.

weighted average of all positional votes of $j \in \mathcal{R}(p)$ for the new position of i, we require p to fulfill the following requirements in order to be a good representative:

- The graph-theoretic distances to i from all $j \in \mathcal{R}(p)$ should be similar to d_{ip}
- The positions of $j \in \mathcal{R}(p)$ in X should be well distributed in close proximity around p.

We propose to construct the partitioning induced by $\mathcal{R}$ only based on the graph structure, not on the layout space, and associate each node $v \in V$ with $\mathcal{R}(p)$ of the closest pivot subject to their graph-theoretic distance. As our algorithm incrementally constructs $\mathcal{R}$, ties are broken by favoring the currently smallest partition. Since all nodes in $\mathcal{R}(p)$ are at least as close to p as to any other pivot, and consequently in the stress drawing, it is appropriate to assume that both conditions are met, cf. Fig. 1.

Even if the positional vote of each pivot is optimal w.r.t. $\mathcal{R}(p)$, it is still not enough to approximate the full stress model. In the full stress model, the iterative algorithm to minimize the stress moves one node at a time while fixing the rest. By setting node i's position in dimension α to

$$X_i^\alpha = \frac{\sum_{j \neq i} w_{ij}(X_j^\alpha + d_{ij}(X_i^\alpha - X_j^\alpha)/||X_i - X_j||)}{\sum_{j \neq i} w_{ij}},$$

it can be shown that the stress monotonically decreases [18]. However, in our model we move node i according to

$$X_i^\alpha = \frac{\sum\limits_{j \in N(i)} w_{ij}\left(X_j^\alpha + \frac{d_{ij}(X_i^\alpha - X_j^\alpha)}{||X_i - X_j||}\right) + \sum\limits_{p \in \mathcal{P} \setminus N(i)} w'_{ip}\left(X_p^\alpha + \frac{d_{ip}(X_i^\alpha - X_p^\alpha)}{||X_i - X_p||}\right)}{\sum\limits_{j \in N(i)} w_{ij} + \sum\limits_{p \in \mathcal{P} \setminus N(i)} w'_{ij}}. \quad (4)$$

Algorithm 1: Sparse Stress

Input: Graph $G = (V, E)$ with $w : E \to \mathbb{R}_{>0}$, and k number of pivots.
Output: 2-dimensional layout $X \in \mathbb{R}^{n \times 2}$

1 sample $\mathcal{P}$ with $|\mathcal{P}| = k$
2 calculate $\mathcal{R}$, all adapted weights w'_{ip}, and all d_{ip} via Alg. 2
3 $X \leftarrow$ `PivotMDS(G)` [4]
4 rescale X such that $\sum_{\{i,j\}\in E} ||X_i - X_j|| = \sum_{\{i,j\}\in E} w_{ij}$
5 **while** *relative change in Eq.* (3) $> 10^{-4}$ **do**
6 **foreach** $i \in V$ **do**
7 **foreach** *dimension* α **do**
8 $t^\alpha \leftarrow \frac{\sum_{j\in N(i)} w_{ij}\left(X_j^\alpha + \frac{d_{ij}(X_i^\alpha - X_j^\alpha)}{||X_i - X_j||}\right) + \sum_{p\in\mathcal{P}\setminus N(i)} w'_{ip}\left(X_p^\alpha + \frac{d_{ip}(X_i^\alpha - X_p^\alpha)}{||X_i - X_p||}\right)}{\sum_{j\in N(i)} w_{ij} + \sum_{p\in\mathcal{P}\setminus N(i)} w'_{ij}}$
9 $X_i \leftarrow t$

This implies that in order to find the globally optimal position of i, we also have to find weights w'_{ip} such that $\frac{w'_{ip}}{\sum_{j\in N(i)} w_{ij} + \sum_{p\in\mathcal{P}\setminus N(i)} w'_{ip}} \approx \frac{\sum_{j\in\mathcal{R}(p)\setminus N(i)} w_{ij}}{\sum_{i\neq j} w_{ij}}$. Since our goal is only to reconstruct the proportions, and our model only knows the shortest path distance between all nodes $i \in V$ and $p \in \mathcal{P}$, we set $w'_{ip} = s/d_{ip}^2$ where $s \geq 1$. At first glance, setting $s = |\mathcal{R}(p)|$ seems appropriate since p represents $|\mathcal{R}(p)|$ addends of the stress model. Nevertheless, this strongly overestimates the weight of close partitions. Therefore, we propose to set $s = |\{j \in \mathcal{R}(p) : d_{jp} \leq d_{ip}/2\}|$. This follows the idea that p is only a good representative for the nodes in $\mathcal{R}(p)$ that are at least as close to p as to i. Since the graph-theoretic distance between i and $j \in \mathcal{R}(p)$ is unknown, our best guess is that j lies on the shortest path from p to i. Consequently, if $d_{jp} \leq d_{ip}/2$, node j must be at least as close to p as to i. Note that $w'_{pp'}$ does not necessarily equal $w'_{p'p}$ for $p, p' \in \mathcal{P}$, and if $k = n$ our model reduces to the full stress model.

Asymptotic running time: To minimize Eq. (3) in each iteration we displace all nodes $i \in V$ according to Eq. (4). Since this requires $|N(i)|+k$ constant time operations, given that all graph-theoretic distances are known, the total time per iteration is in $\mathcal{O}(kn + m)$. Furthermore, only the distances between all $i \in V$ and $p \in \mathcal{P}$ have to be known which can be done in $\mathcal{O}(k(m + n \log n))$ time and requires $\mathcal{O}(kn)$ additional space. If the graph-theoretic distances for all $p \in \mathcal{P}$ are computed with a multi-source shortest path algorithm (MSSP), it is possible to construct $\mathcal{R}$ as well as calculate all w'_{ip} during its execution without increasing its asymptotic running time, see Alg. 2. The full algorithm to minimize our sparse stress model is presented in Alg. 1.

Algorithm 2: Multi-Source Shortest Path

Input: Graph $G = (V = \{0, \ldots, n-1\}, E), w : E \to \mathbb{R}_{>0}$, and pivots $\{p_0, \ldots, p_{k-1}\}$.
Data: Priority-Queue Q containing dummy element $(k \cdot n, \infty)$
Output: distances d_{ip} and weights w'_{ip}

```
1  for 0 ≤ i < k do pP(i) ← 0; pD(i) ← ∅; upsert(Q, i·n + p_i, 0);
2  oDist ← 0; tA ← ∅; tC ← ∅;
3  while Q not empty do
4      cInd, cDist ← pop(Q);
5      if oDist ≠ cDist then
6          assign nodes in tA to pivot of smallest region
7          for each new node in the region of p_i do push(pD(i), oDist)
8          for index ∈ tC do
9              pInd ← ⌊cInd/n⌋; v ← cInd − pInd·n; p ← p_pInd;
10             if v and p not adj. then
11                 d_{v,p} = oDist; w'_{v,p} = pP(pInd)/(oDist·oDist);
12         for 0 ≤ i ≤ k do
13             while pD(i)_{pP(i)} ≤ cDist/2 do pP(i) ← pP(i) + 1
14         oDist = cDist; tA ← ∅; tC ← ∅;
15     if cInd = dummy then continue
16     mark cInd; tC ← tC ∪ {cInd};
17     pInd ← ⌊cInd/n⌋; v ← cInd − pInd·n;
18     if v not assigned to region then tA ← tA ∪ {cInd}
19     for w ∈ N(v) do
20         wInd ← cInd − v + w
21         if wInd not marked then upsert(Q, wInd, cDist + w({v, w}))
```

4 Experimental Evaluation

We report on two sets of experiments. The first is concerned with the evaluation of the impact of different pivot sampling strategies. The second set is designed to assess how well the different sparse stress models approximate the full stress model, in both absolute terms and in relation to the speed-up achieved.

For the experiments, we implemented the sparse stress model, Alg. 1, as well as different sampling techniques in Java using Oracle SDK 1.8 and the yFiles 2.9 graph library (`www.yworks.com`).[1] The tests were carried out on a single 64-bit machine with a 3.60GHz quad-core Intel Core i7-4790 CPU, 32GB RAM, running Ubuntu 14.10. The reported running times were averaged over 25 iterations and measured using the `System.currentTimeMillis()` command. We note here that all drawing algorithms, except stated otherwise, were initialized with a 200 PivotMDS layout [4]. Furthermore, the maximum number of itera-

[1] A stand-alone version is available at `https://github.com/MarkOrtmann/sparse-stress`.

Table 1: Dataset: n, m, $\delta(G)$, $\Delta(G)$, and $D(G)$ denote the number of nodes, edges, the min. and max. degree, and the diameter, respectively. Column $\{deg(i)\}$ and $\{d_{ij}\}$ show the degree and distance distribution, respectively. Bipartite graphs are marked with * and weighted graphs with **.

graph	n	m	$\delta(G)$	$\Delta(G)$	$D(G)$	$\{deg(i)\}$	$\{d_{ij}\}$
dwt1005	1005	3808	3	26	34		
1138bus	1138	1458	1	17	31		
plat1919	1919	15240	2	18	43		
3elt	4740	13722	3	9	65		
USpowerGrid	4941	6594	1	19	46		
commanche	7920	11880**	3	3	438.00		
LeHavre	11730	15133**	1	7	33800.67		
pesa	11738	33914	2	9	208		
bodyy5	18589	55346	2	8	132		
finance256	20657	71866	1	54	55		
btree (binary tree)	1023*	1022	1	3	18		
qh882	1764*	3354	1	14	32		
lpship04l	2526*	6380	1	84	13		

tions for the full stress algorithm was set to 500. As stress is not resilient against scaling, see Eq. (1), we optimally rescaled each drawing such that it creates the lowest possible stress value [2].

Data: We conducted our experiments on a series of different graphs, see Tab. 1, most of them taken from the sparse matrix collection [9]. We selected these graphs as they differ in their structure and size, and are large enough to compare the results of different techniques. Two of the graphs, *LeHavre* and *commanche*, have predefined edge lengths that were derived from the node coordinates. We did not modify the graphs in any way, except for those that were disconnected, in which case we only kept the largest component.

4.1 Sampling Evaluation

In Section 3 we discussed how vital the proper selection of the pivots is for our model. In the optimal case we would sample pivots that are well distributed over the graph, creating regions of equal complexity, and are central in the drawing of their regions. In order to evaluate the impact of different sampling strategies on the quality of our sparse stress model and recommend a proper sampling scheme, we compared a set of different strategies:

- *random*: nodes are selected uniformly at random
- *MIS filtration*: nodes are sampled according to the maximal independent set filtration algorithm by Gajer et al. [15]. Once $n \leq k$, the coarsening

stops. If $n < k$, unsampled nodes from the previous level are randomly added

- *max/min Euclidean*: starting with a uniform, randomly chosen node, $\mathcal{P}$ is extended by adding $\arg\max_{i \in V \setminus \mathcal{P}} \min_{p \in \mathcal{P}} ||x_i - x_p||$

- *max/min sp*: similar to *max/min Euclidean*, except that $\mathcal{P}$ is extended according $\arg\max_{i \in V \setminus \mathcal{P}} \min_{p \in \mathcal{P}} d_{ip}$ [4]

Pretests showed that the *max/min sp* strategy initially favors sampling leaves, but nevertheless produces good results for large k. Thus, we also evaluated strategies building on this idea, while trying to overcome the problem of leaf node sampling.

- *max/min random sp*: similar to *max/min sp*, but each node i is sampled with a probability proportional to $\min_{p \in \mathcal{P}} d_{ip}$

- *k-means layout*: the nodes are selected via a k-means algorithm, running at most 50 iterations, on the initial layout

- *k-means sp*: initially k nodes with *max/min sp* are sampled, succeeded by k-means sampling using the shortest path entries of these pivots

- *k-means + max/min sp*: $\mathcal{P}$ is initialized with $k/2$ pivots via *k-means layout* and the remaining nodes are sampled via *max/min sp*

Using the k-means algorithm comes with a problem since the representative computed for each of the k regions, the so-called centroid, is an artificial data point. Therefore, after every single iteration of the algorithm we replace each centroid by that node in its region which has the smallest (Euclidean) distance. This is a reasonable replacement strategy, as the position of a centroid equals the arithmetic mean position of the points in its region.

To quantify how well suited each of the sampling techniques is for our model, we ran each combination on each graph with $k \in \{50, 51, \ldots, 200\}$ pivots. For all tests we forced termination of the sparse stress algorithm after 200 iterations if it did not converge before. Since all techniques at some point rely on a random decision, we repeated each execution 20 times in order to ensure we do not rest our results upon outliers. To distinguish the applicability of the different techniques to our model, we used two measures. The first measure is the normalized stress which is the stress value divided by $\binom{n}{2}$. While the normalized stress assesses the quality of our drawing, we also calculated the Procrustes statistic [8, 33] which measures how well the layout matches the full stress drawing. More precisely, the Procrustes statistic $R^2(X, Y) \in [0, 1]$ is the normalized sum of squared (Euclidean) distances of the node positions in layout X and the (ideal) reference layout Y. In order to minimize the Procrustes statistic the layout X is transformed under scaling, translation, and rotation to match Y as best as possible respective $R^2(X, Y)$. This implies that a low

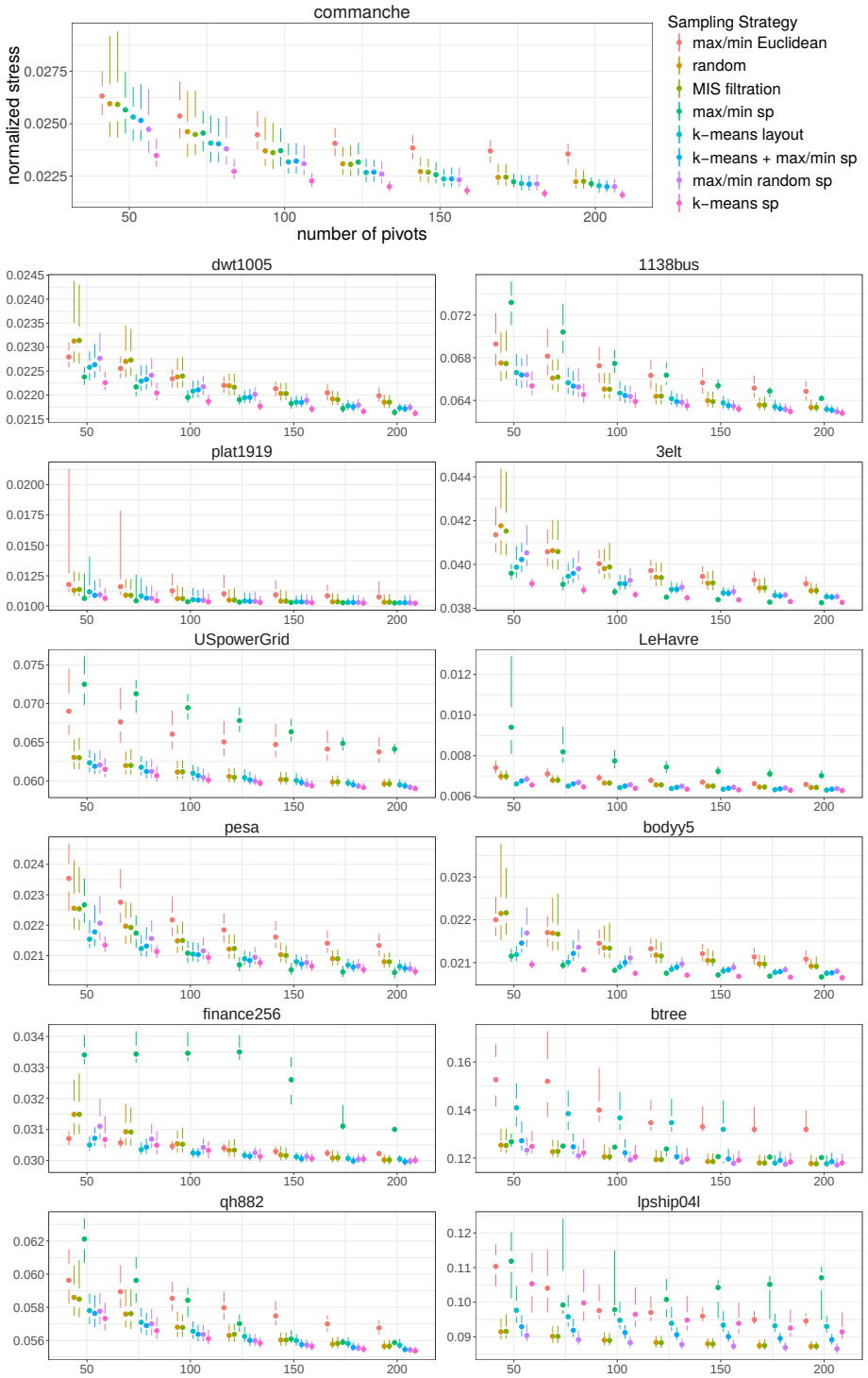


Figure 2: Comparison of different sampling strategies and number of pivots w.r.t. the resulting normalized stress value.

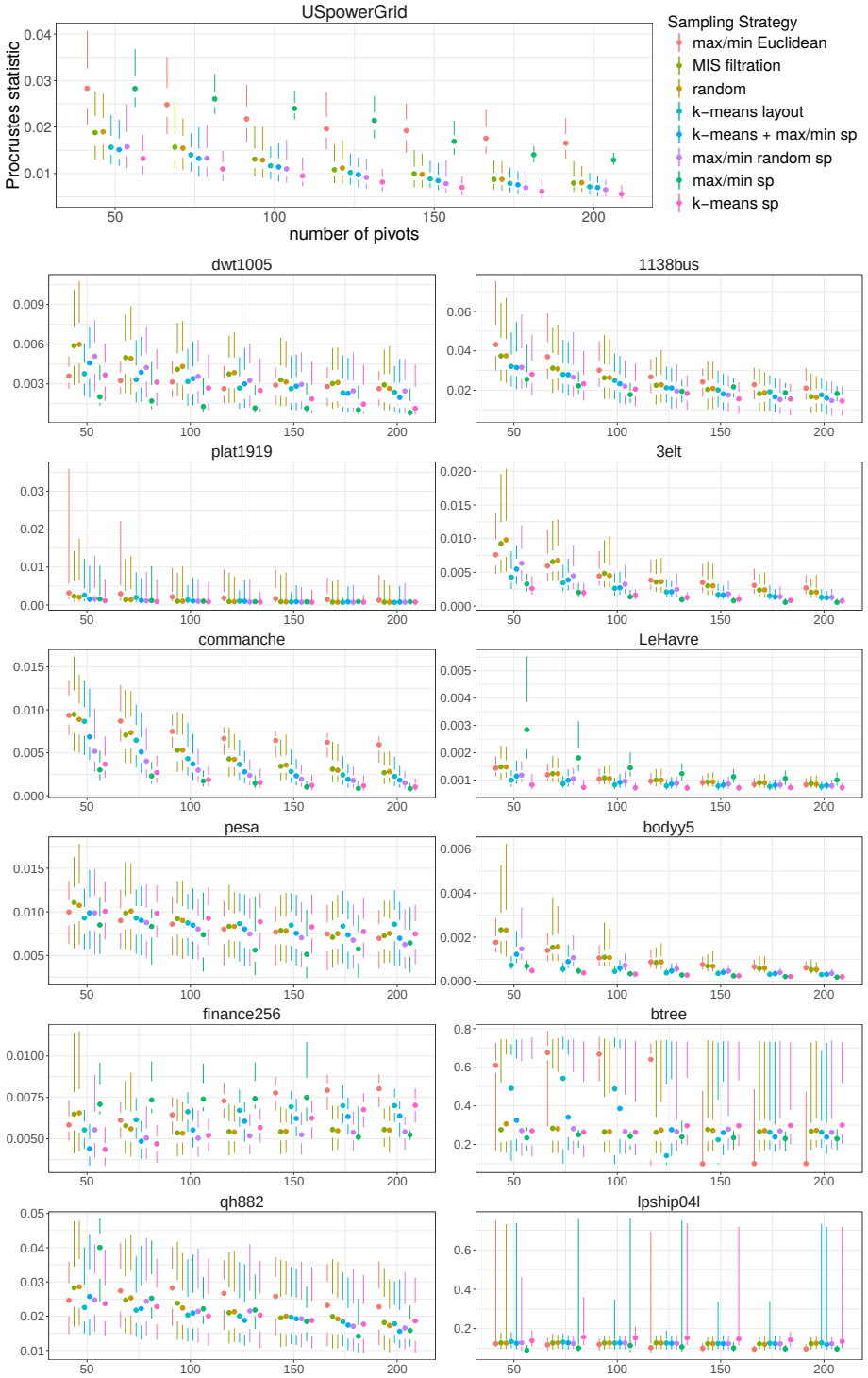
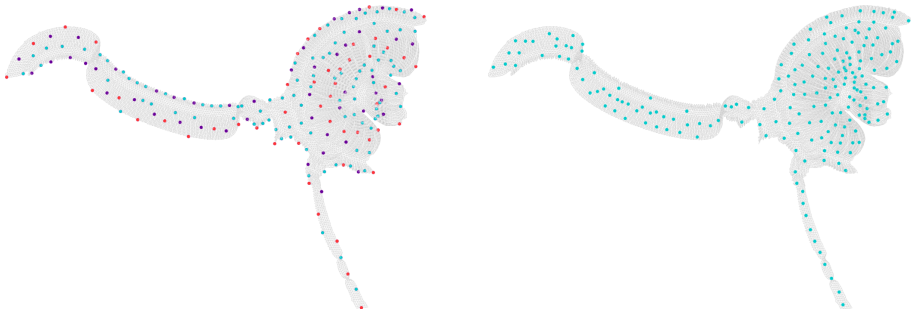


Figure 3: Comparison of different sampling strategies and number of pivots w.r.t. the Procrustes statistic.



max/min sp *k-means sp*

Figure 4: Pivots sampled by (left) *max/min sp* and (right) *k-means sp* for *pesa*. While the first 50 pivots sampled (red) by *max/min sp* mostly lie on the contour, already for 100 pivots (red+purple), the pivots lie central in the left arm and for $k = 200$ (red+purple+cyan) the pivots are well distributed all over the arm. In comparison, *k-means sp* for $k = 200$ still mainly samples pivots in the left arm that are central in the layout.

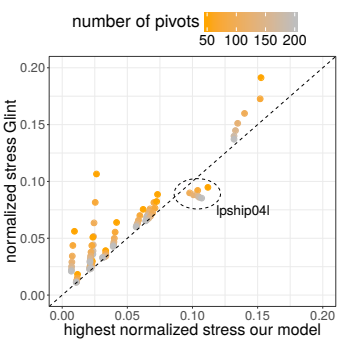
value $R^2(X, Y)$ indicates that X and Y are very similar. The equation for the Procrustes statistic incorporating the transformation of X [8] is given by

$$R^2(X, Y) = 1 - \frac{[\mathrm{tr}(X^T Y Y^T X)^{\frac{1}{2}}]^2}{\mathrm{tr}(X^T X)\,\mathrm{tr}(Y^T Y)}$$

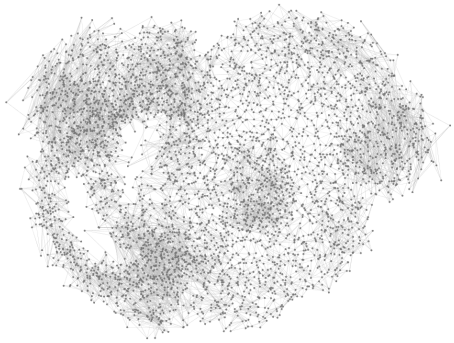
with $\mathrm{tr}(A)$ denoting the trace of the squared matrix A and $A^{\frac{1}{2}}$ being the square root of A.

The results of these experiments are presented in Figs. 2 and 3. Each dot in these plots represents the median and each line starts at the 25%, 75% percentile and ends at the 5%, 95% percentile, respectively. For the sake of readability we binned each 25 consecutive sample sizes. Furthermore, the strategies were ordered according to their overall ranking w.r.t. the evaluated measure. Therefore, the ordering summarizes the overall performance of each strategy for the given measure (left plot / top legend lowest performance; right plot / bottom legend highest performance). For most of the graphs, using *k-means sp* sampling yields the layouts with the lowest normalized stress value. There are only two graphs where this strategy performs worse than other tested strategies. The one graph where *k-means sp* is outclassed, albeit only for large k by *max/min sp*, is *pesa*. The reason for this result is that *k-means sp* mainly samples pivots in the center of the left arm creating twists, see Tab. 6. *Max/min sp* for small k in contrast mostly samples nodes on the contour of the arm, but once k reaches a certain threshold, the resulting distribution of the pivots prevents twists, yielding a lower normalized stress value, see Fig. 4.

The explanation for the poor behavior for *lpship04l* is strongly related to its structure. The low diameter of 13 causes, after a few iterations, the *max/min sp* strategy to repeatedly sample nodes that are part of the same cluster, see



(a) Normalizes stress Glint vs. highest normalized stress our model.



(b) *3elt* drawn using Glint with 200 pivots.

Figure 5: Comparison between Glint [24] and our sparse stress model. In (a) the (median) normalized stress of Glint is plotted against that of our model. Each point corresponds to a single graph of Tab. 1 in combination with the number of pivots ($[50, 200]$ binned each 25 consecutive pivot sizes) used to calculate the drawing. A point lying above the main diagonal (black dashed line) means that our approach creates a drawing with lower stress than Glint even when using the lowest performing sampling strategy respective the graph-pivot combination. The drawing of *3elt* created by Glint using 200 pivots is shown in (b).

Tab. 6, and consequently are structurally very similar. As *k-means sp* builds on *max/min sp*, it can only slightly improve the pivot distribution. The argument that the problem is related to the structure is reinforced by the outcome of the *random* strategy. Still, except for these two graphs, *k-means sp* generates the best outcomes, and since this strategy is also strongly favorable over the others subject to the Procrustes statistics, see Fig. 3, our following evaluation always relies on this sampling strategy. However, we note that the Procrustes statistic for *btree* and *lpship04l* are by magnitudes larger than for any other tested graph. While for *lpship04l* this is mostly caused by the quality of the drawings, this is only partly true for *btree*. The other factor contributing to the high Procrustes statistic for *btree* is caused by the restricted set of operations provided by the Procrustes analysis. As scaling, translation, and rotation are used to find the best match between two layouts, the Procrustes analysis cannot resolve reflections. Therefore, if in one layout of *btree*, the subtree T_1 of v is drawn to the right of subtree T_2 of v and vice versa in the other drawing, although the two layouts are identical, the statistic will be high. This symmetry problem mainly explains the low performance w.r.t. *btree*.

Recall that the Glint framework [24] samples the set of pivots independently and uniformly at random, for each node. As a result, this technique requires in the worst-case an APSP computation and therefore is not a sparse stress model in the narrow sense of the definition. Fig. 5(a) shows a comparison between our model and the Glint model. Each point in the figure corresponds to a

single graph of Tab. 1 in combination with the (binned) number of pivots used to calculate the drawing. The y-axis shows the (median) normalized stress of Glint and the x-axis the (median) normalized stress of our model when using the lowest performing sampling strategy for the given graph - number of pivots combination. A point lying above the main diagonal indicates that our approach creates a drawing with lower stress than Glint. As all points except for *lpship04l* lie above the main diagonal, we can see that our approach is not only applicable to large graphs, but also that selecting pivots based on the graph's structure is favorable over random assignments. Furthermore, comparing Fig. 5(b) to the drawing obtained by our approach for *3elt*, cf. Tab. 6, we can see that Glint's random pivot assignment strategy creates blurred drawings hiding the otherwise clearly visible graph structure.

4.2 Full Stress Layout Approximation

The next set of experiments is designed to assess how well our sparse stress model using *k-means sp* sampling, as well as related sparse stress techniques, resemble the full stress model. For this we compared the median stress layout over 25 repetitions on the same graph of our sparse stress model with $k \in \{50, 100, 200\}$, with MARS,[2] maxent,[3] PivotMDS, 1-stress, and the weighted version of GRIP.[4] The number of iterations of our model as well as for MARS and 1-stress have been limited to 200. Furthermore, we tested MARS with 100 and 200 pivots and report the layout with the smallest stress from the drawings obtained by running `mars` with argument `-p` $\in \{1, 2\}$ combined with a PivotMDS or randomly initialized layout.

Besides comparing the resulting stress values and Procrustes statistics, we compared the distribution of pairwise Euclidean distances subject to their graph-theoretic distances. Since, as mentioned in the previous subsection, the Procrustes statistic cannot handle reflective symmetries, we propose to evaluate the similarity of the sparse stress layouts with the full stress layout via Gabriel graphs [14]. The Gabriel graph $GG(X)$ of a given layout X contains an edge between a pair of points if and only if the disc associated with the diameter of the endpoints does not contain any other point. Since the treatment of identical positions, i.e., nodes with identical coordinates in the layout, is not defined for Gabriel graphs, we resolve this by adding edges between each pair of identical positions. We assess the similarity between the Gabriel graph of the full stress layout (X) and the sparse stress layouts (Y) by comparing the l-neighborhoods of a node in the graphs using the Jaccard coefficient. More formally the l-neighborhood Gabriel graph similarity of a node $v \in V$ is defined as $\frac{|N_{GG(X)}(v,l) \cap N_{GG(Y)}(v,l)|}{|N_{GG(X)}(v,l) \cup N_{GG(Y)}(v,l)|} \in [0, 1]$ with $N_G(v, l) = \{w \in V \setminus \{v\} : d_{G,vw} \leq l\}$.

[2] `https://github.com/marckhoury/mars`

[3] We are grateful to Yifan Hu for providing us with the code.

[4] `http://www.cs.arizona.edu/~kobourov/GRIP/`

Table 2: Stress and Procrustes statistics: sparse model values are written in **bold** when no larger than minimum over previous methods.

graph	full stress	sparse 200	sparse 100	sparse 50	maxent	MARS 200	MARS 100	GRIP	1-stress	PivotMDS
stress										
dwt1005	10 729	**10 940**	**11 081**	**11 329**	21 623	17 660	20 134	52 517	12 495	14 459
1138bus	39 974	**40 797**	**41 471**	**42 686**	44 650	64 363	63 614	54 986	73 512	75 427
plat1919	18 572	**18 840**	**19 072**	**19 719**	23 850	53 246	64 166	113 765	75 973	82 865
3elt	422 940	**426 564**	**430 200**	**437 051**	585 967	503 600	754 134	934 206	555 934	634 401
USpowerGrid	702 055	**720 642**	**731 187**	**749 464**	1 021 457	766 535	783 888	1 495 373	1 111 216	1 123 698
commanche	654 694	**677 220**	**699 890**	**749 609**	1 507 654	2 761 605	3 145 489	1 539 767	2 085 818	2 157 943
LeHavre	439 188	**433 030**	**441 986**	**454 785**	1 231 283	12 012 307	12 570 692	8 658 371	1 255 474	1 305 577
pesa	1 373 514	**1 417 449**	**1 452 975**	**1 495 512**	10 423 779	3 563 772	8 281 116	2 957 738	3 486 176	3 325 889
bodyy5	3 547 659	**3 566 636**	**3 585 087**	**3 630 380**	5 248 755	6 385 559	4 072 905	10 389 846	4 245 006	4 715 728
finance256	6 175 210	**6 415 761**	**6 474 787**	**6 582 890**	8 151 335	7 267 598	8 643 239	19 817 355	12 257 268	11 380 089
btree	60 206	**61 839**	**63 325**	**66 122**	67 871	103 436	100 767	96 235	157 988	164 329
qh882	84 524	**86 345**	**87 695**	**89 556**	103 601	117 195	161 113	127 914	146 935	143 142
lpship04l	250 599	**297 547**	**316 674**	343 694	329 255	558 923	542 667	771 284	775 813	793 238
Procrustes statistic										
dwt1005		**0.001**	**0.005**	**0.003**	0.027	0.008	0.018	0.263	0.004	0.008
1138bus		**0.009**	**0.016**	0.025	0.022	0.148	0.145	0.071	0.097	0.102
plat1919		**0.000**	**0.000**	**0.001**	0.015	0.026	0.031	0.236	0.045	0.051
3elt		**0.001**	**0.001**	**0.002**	0.026	0.009	0.029	0.199	0.017	0.023
USpowerGrid		**0.006**	**0.008**	**0.012**	0.068	0.014	0.018	0.256	0.051	0.051
commanche		**0.001**	**0.002**	**0.005**	0.039	0.026	0.167	0.092	0.066	0.066
LeHavre		**0.001**	**0.001**	**0.001**	0.012	0.163	0.173	0.256	0.010	0.010
pesa		**0.009**	**0.010**	**0.010**	0.095	0.025	0.070	0.017	0.021	0.021
bodyy5		**0.000**	**0.000**	**0.000**	0.012	0.011	0.003	0.100	0.004	0.007
finance256		0.009	**0.006**	**0.005**	0.013	0.007	0.018	0.206	0.042	0.041
btree		0.748	**0.165**	0.241	0.233	0.360	0.367	0.386	0.361	0.364
qh882		**0.015**	**0.015**	**0.021**	0.046	0.061	0.114	0.075	0.086	0.079
lpship04l		0.176	**0.112**	**0.148**	0.160	0.246	0.587	0.463	0.393	0.401

Table 3: Runtime in seconds: fastest sparse model yielding lower stress than best previous method, cf. Tab. 2, is written in **bold**. Times of implementations written in `C/C++` (marked with *) measured via `clock()` command.

graph	full stress	sparse 200	sparse 100	sparse 50	maxent*	MARS 200*	MARS 100*	GRIP*	1-stress	Pivot MDS
dwt1005	1.26	0.33	0.15	**0.09**	0.47	1.02	2.36	0.06	0.08	0.06
1138bus	2.20	0.41	0.16	**0.09**	0.91	3.16	1.96	0.20	0.06	0.04
plat1919	9.70	1.00	0.45	**0.24**	1.15	6.80	4.79	0.19	0.31	0.20
3elt	31.82	2.28	0.93	**0.43**	2.26	16.31	8.43	0.71	0.37	0.23
USpowerGrid	36.48	1.85	0.67	**0.37**	2.53	13.54	7.62	1.67	0.28	0.21
commanche	340.10	10.78	3.63	**1.51**	3.60	22.72	12.43	2.29	0.47	0.35
LeHavre	475.05	12.75	4.90	**2.19**	6.31	27.57	19.50	10.18	0.81	0.54
pesa	373.23	9.61	4.14	**1.50**	5.96	50.10	42.68	3.56	0.95	0.60
bodyy5	463.47	12.53	4.31	**2.01**	9.97	46.63	9.27	10.43	1.64	1.04
finance256	1016.92	10.44	4.27	**2.28**	14.76	32.16	24.66	12.12	2.51	1.60
btree	7.79	0.42	0.18	**0.09**	0.63	2.70	1.48	0.06	0.06	0.03
qh882	6.61	0.65	0.28	**0.15**	0.97	8.45	5.79	0.15	0.17	0.14
lpship04l	18.30	0.73	**0.31**	0.18	0.99	7.06	7.63	0.16	0.15	0.10

A further measure we introduce evaluates the visual error. More precisely, we measure for a given node v the percentage of nodes that lie in the drawing area of the l-neighborhood, but are not part of it. We calculate this value by computing the convex hull induced by the l-neighborhood and then test for each other node if it belongs to the hull or not. This number is then divided by $n - |\{w \in V : d_{vw} \leq l\}|$. Therefore, a low value implies that there are only a few nodes lying in the region while high values indicate we cannot distinguish non l-neighborhood and l-neighborhood nodes in the drawing. This measure is to a certain extent similar to the precision of neighborhood preservation [17]. Let $CH(N'_G(v,l))$ denote the convex hull of $v \in V$ induced by $N'_G(v,l) = \{v\} \cup N_G(v,l)$ in the layout X. Then the distance l visual error of v is given by

$$\frac{|\{w \in V \setminus N'_G(v,l) : w \in CH(N'_G(v))\}|}{n - |N'_G(v,l)|} \in [0,1].$$

Note that for this evaluation we always calculated $N_G(v,l)$ w.r.t. the unweighted shortest-path distances.

The results of all these experiments, see Tabs. 2 and 6, and Figs. 6 and 7, reveal that our model is more adequate in resembling the full stress drawing than any other of the tested algorithms, while showing comparable running times that scale nicely with k, cf. Tab. 3. The error plots in Tab. 6 expose the strength of our scheme. We can see that, while all approaches work very well in representing short distances, our approach is more precise in approximating middle and especially long distances of the full stress model, explaining our good results. As the evaluation clearly shows that our approach yields better approximations of the full stress model, we rather want to discuss the low performance of our model for *lpship04l* and thereby expose one weakness of our approach.

Looking at the sparse 50 drawing of *lpship04l* in Tab. 6, we can see that a large portion of nodes has a similar or even the same position. This is because *lpship04l* has a lot of nodes that share very similar graph-theoretic distance vectors, exhibit highly overlapping neighborhoods, and are drawn in close proximity

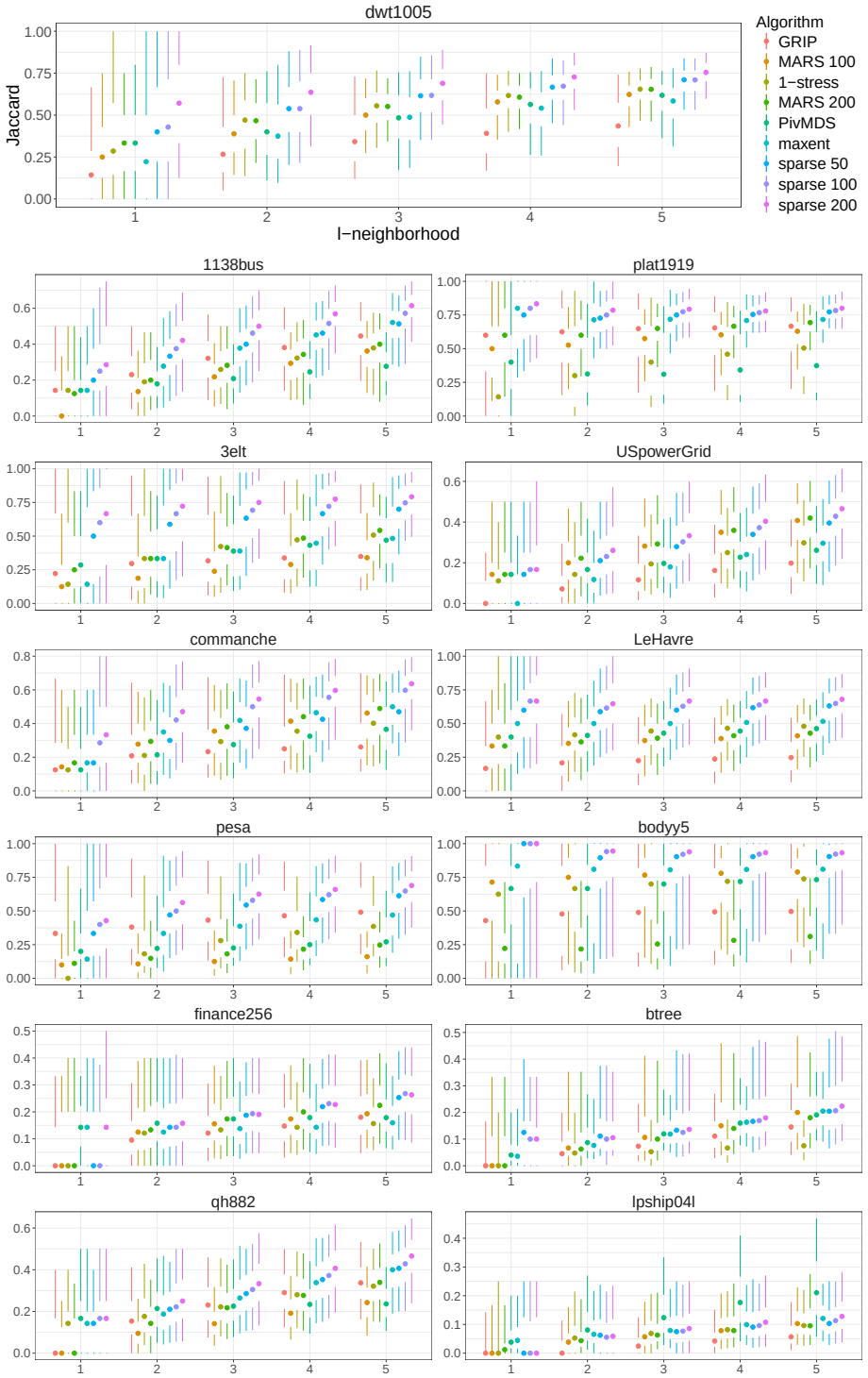


Figure 6: The similarity of the Gabriel graph of the full stress layout and the Gabriel graph of the layout algorithms under consideration as a function of l. For each node of the graph the l-neighborhood in the Gabriel graph of the full stress layout and the layout algorithm are compared by calculating the Jaccard coefficient. A higher value indicates that the nodes share a high percentage of common neighbors in the different Gabriel graphs.

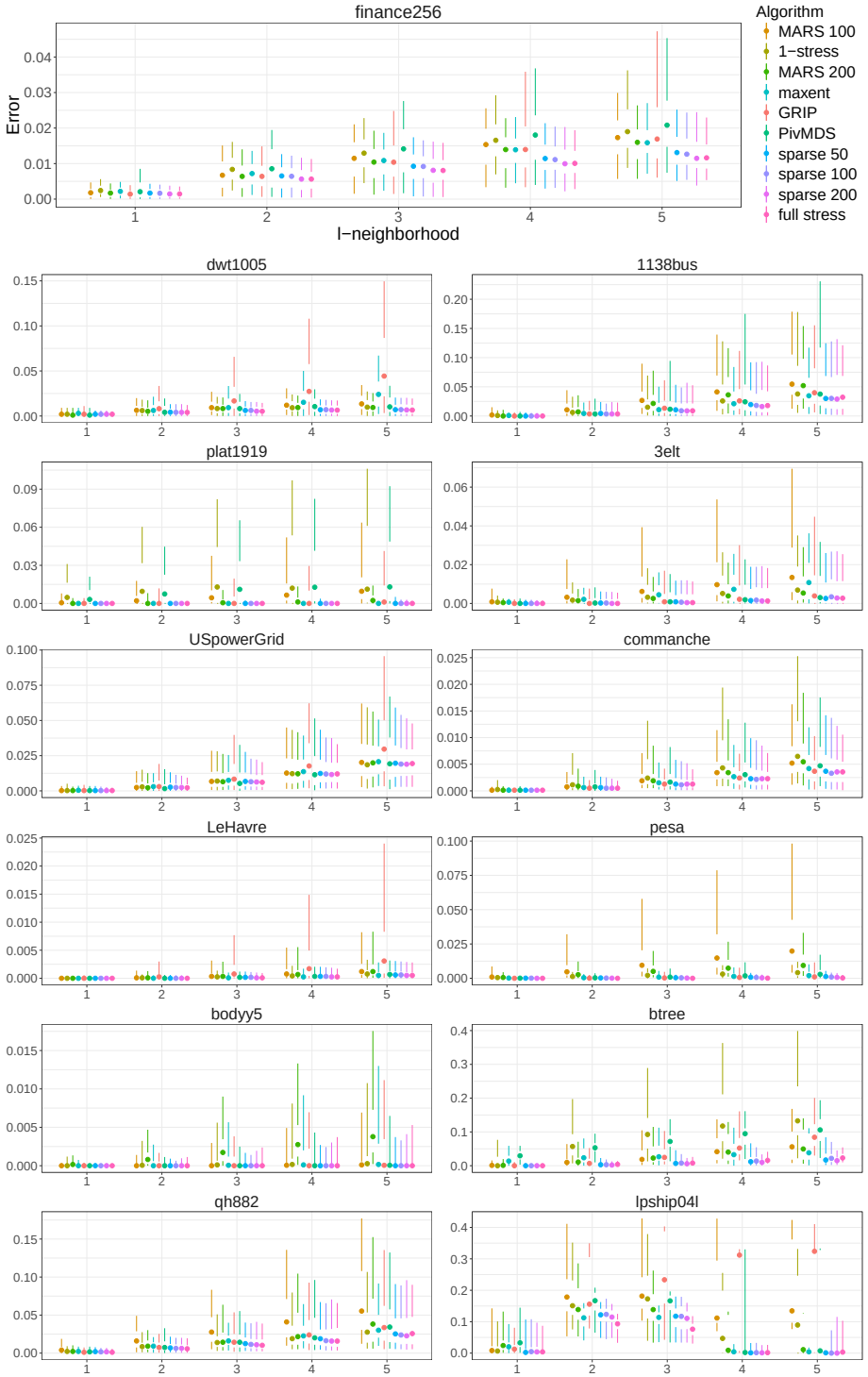


Figure 7: Error charts as a function of l. For each node of the graph, the convex hull w.r.t. the coordinates of the nodes in the l-neighborhood is computed. For each of the convex hulls the error is calculated by counting the number of non l-neighborhood nodes that lie inside or on the contour of this hull divided by their total number.

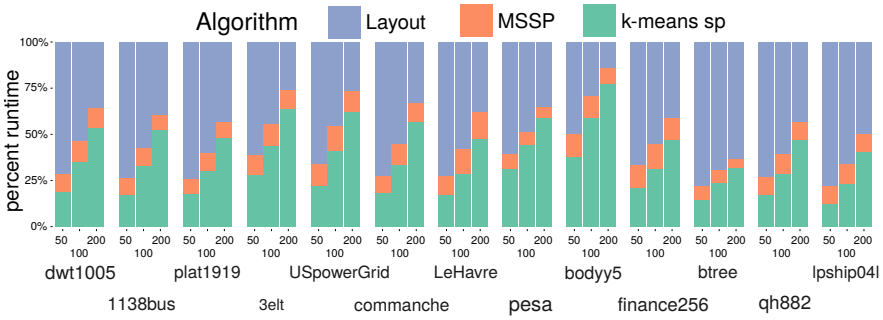


Figure 8: Running time composition of sparse 50, sparse 100, sparse 200 with *k-means sp* ($c = k$) respective the times shown in Tab. 5. For increasing number of pivots, the time of *k-means sp* with $c = k$ starts dominating the overall running time of the sparse stress model.

in the initial PivotMDS layout. While our model would rely on small variations of the graph-theoretic distances to create a good drawing, we diminish these differences even further by restricting our model to $\mathcal{P}$. Consequently, the positional vote for two similar non-pivot nodes i and j that lie in the same partition will only slightly differ, mainly caused by their distinct neighbors. However, as these neighbors are also in close proximity in the initial drawing of *lpship04l*, the distance between i and j will not increase. Therefore, if the graph has a lot of structurally very similar nodes and the initial layout has poor quality, our approach will inevitably create drawings where nodes are placed very close to one another. Note that this also explains the good performance of Glint compared to our model for *lpship04l*, see Fig. 5(a).

4.3 Runtime Improvement

While *k-means sp* is preferable over other sampling techniques, as shown in Figs. 2 and 3, it has one serious drawback. Since the input for the k-means algorithm used by this sampling strategy is an $n \times k$ matrix, the running time complexity is $\mathcal{O}(nk^2)$. This implies that the preprocessing time of our sparse stress model using *k-means sp* is $\mathcal{O}(\max\{nk^2, k(m + n \log n)\})$. Consequently, at some point the running time of the sparse stress model is entirely dominated by *k-means sp*. The composition of the running times (Tab. 5) shown in Fig. 8 draws a clear picture, namely that already for 100/200 pivots the sparse stress model mostly spends 25%/50% of its overall running time for the sampling via *k-means sp*.

The simplest way to resolve this issue is to sample only a constant number, c, shortest-path entries via *max/min sp* and then use this $n \times c$ matrix as input for the k-means algorithm. We will in the following show that setting $c = 25$ clearly reduces the running time, while the results compared to the $c = k$ version of *k-means sp* used in the above evaluation stay approximately the same.

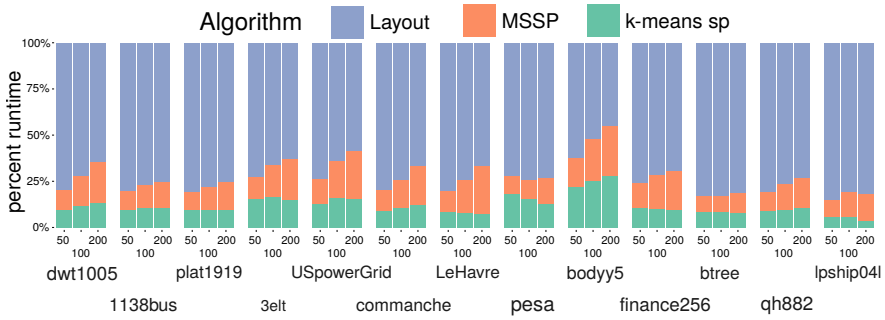


Figure 9: Running time composition of sparse 50, sparse 100, sparse 200 with *k-means sp* ($c = 25$) respective the times shown in Tab. 5. Increasing the number of pivots does not affect the share of the total running w.r.t. *k-means sp* with $c = 25$.

In order to evaluate the impact of setting $c = 25$, we reran the exact same evaluation process using the above given parameters and graphs. Looking at Fig. 9, we can see that fixing the number of features stabilizes the share of the total running time w.r.t. *k-means sp*. Furthermore, Fig. 9 shows that minimizing Eq. (3) takes most of the time, yet as the number of pivots raises the share of MSSP increases. We note that for *bodyy5* the portion of minimizing Eq. (3) is comparably small, as Alg. 1 converges after only a small number of iterations.

Tabs. 4 and 5 show the results w.r.t. stress, Procrustes statistic and the running time in seconds using *k-means sp* with $c = k$ and $c = 25$. Looking at the running times (Tab. 5) we can see that fixing the number of features reduces the overall running time and the speed-up raises as the number of pivots increases. While this is not a surprising result, taking a closer look at the stress values (Tab. 4) reveals that, except for *btree* and *lpship04l*, the difference in these values is below 0.5%. This and the fact that the Procrustes statistics are also very similar implies that reducing the running time by setting c to a small constant does not necessarily come at the cost of a lower layout quality. Since the results for $c = 25$ are approximately the same as for $c = k$, except for *btree* and *lpship*, we omit showing the results of the evaluation via Gabriel graphs and convex hulls as well as the drawings and distance-error charts. However, it should be emphasized that *k-means sp* with a constant number of features still outperforms all the other considered sampling techniques.

Table 4: Comparison of the stress and Procrustes statistics of the sparse stress model with $k \in \{50, 100, 200\}$ for *k-means sp* sampling using $c = k$ and $c = 25$. The smaller of the two values is written in **bold**.

graph	sparse 200			sparse 100			sparse 50		
stress									
	$c = k$	$c = 25$	difference %	$c = k$	$c = 25$	difference %	$c = k$	$c = 25$	difference %
dwt1005	**10 940**	10 953	0.119	**11 081**	11 112	0.280	11 329	**11 323**	-0.053
1138bus	**40 797**	40 965	0.412	41 471	**41 459**	-0.029	42 686	**42 548**	-0.323
plat1919	**18 840**	18 858	0.096	**19 072**	19 121	0.257	**19 719**	19 780	0.309
3elt	**426 564**	426 621	0.013	**430 200**	430 701	0.116	**437 051**	437 379	0.075
USpowerGrid	**720 642**	721 206	0.078	**731 187**	732 818	0.223	**749 464**	751 848	0.318
commanche	**677 220**	678 432	0.179	**699 890**	700 412	0.075	749 609	**746 150**	-0.461
LeHavre	433 030	**433 000**	-0.007	**441 986**	442 242	0.058	**454 785**	457 175	0.526
pesa	1 417 449	**1 409 833**	-0.537	1 452 975	**1 447 871**	-0.351	1 495 512	**1 492 049**	-0.232
bodyy5	**3 566 636**	3 567 009	0.010	**3 585 087**	3 587 358	0.063	3 630 380	**3 629 886**	-0.014
finance256	6 415 761	**6 391 041**	-0.385	6 474 787	**6 458 748**	-0.248	6 582 890	**6 562 610**	-0.308
btree	**61 839**	63 509	2.701	**63 325**	63 906	0.917	**66 122**	66 993	1.317
qh882	**86 345**	86 397	0.060	87 695	**87 449**	-0.281	**89 556**	89 622	0.074
lpship04l	**297 547**	308 109	3.550	**316 674**	317 765	0.345	**343 694**	350 164	1.882
Procrustes statistic									
	$c = k$	$c = 25$	difference	$c = k$	$c = 25$	difference	$c = k$	$c = 25$	difference
dwt1005	**0.001**	0.003	0.002	0.005	**0.003**	-0.002	**0.003**	**0.003**	0.000
1138bus	**0.009**	0.010	0.001	0.016	**0.011**	-0.005	0.025	**0.019**	-0.006
plat1919	**0.000**	**0.000**	0.000	**0.000**	**0.000**	0.000	**0.001**	**0.001**	0.000
3elt	**0.001**	**0.001**	0.000	**0.001**	**0.001**	0.000	**0.002**	**0.002**	0.000
USpowerGrid	**0.006**	0.007	0.001	**0.008**	**0.008**	0.000	**0.012**	0.013	0.001
commanche	**0.001**	**0.001**	0.000	**0.002**	**0.002**	0.000	0.005	**0.003**	-0.002
LeHavre	**0.001**	**0.001**	0.000	**0.001**	**0.001**	0.000	**0.001**	**0.001**	0.000
pesa	**0.009**	**0.009**	0.000	0.010	**0.008**	-0.002	0.010	**0.009**	-0.001
bodyy5	**0.000**	**0.000**	0.000	**0.000**	**0.000**	0.000	**0.000**	**0.000**	0.000
finance256	0.009	**0.006**	-0.003	**0.006**	**0.006**	0.000	**0.005**	0.006	0.001
btree	0.748	**0.133**	-0.615	**0.165**	0.244	0.079	0.241	**0.208**	-0.033
qh882	**0.015**	**0.015**	0.000	**0.015**	**0.015**	0.000	**0.021**	0.030	0.009
lpship04l	0.176	**0.103**	-0.073	**0.112**	0.125	0.013	0.148	**0.127**	-0.021

Table 5: Comparison of the runtime in seconds of the sparse stress model with $k \in \{50, 100, 200\}$ for *k-means sp* sampling using $c = k$ and $c = 25$. The smaller of the two values is written in **bold**.

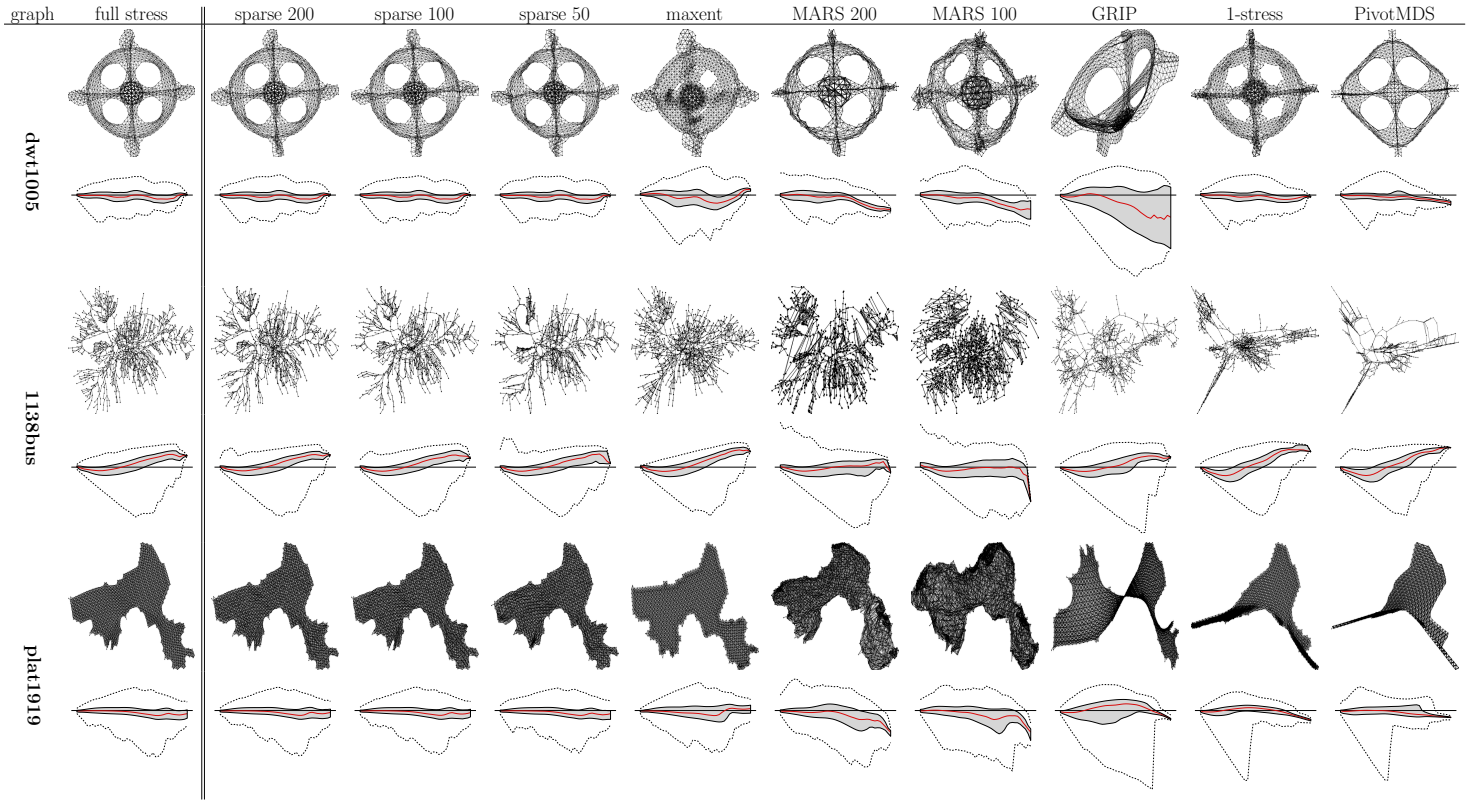
graph	sparse 200			sparse 100			sparse 50		
	$c = k$	$c = 25$	speed-up	$c = k$	$c = 25$	speed-up	$c = k$	$c = 25$	speed-up
dwt1005	0.33	**0.17**	1.94	0.15	**0.11**	1.36	0.09	**0.08**	1.12
1138bus	0.41	**0.25**	1.64	0.16	**0.13**	1.23	0.09	**0.08**	1.12
plat1919	1.00	**0.59**	1.69	0.45	**0.34**	1.32	0.24	**0.22**	1.09
3elt	2.28	**1.04**	2.19	0.93	**0.58**	1.60	0.43	**0.38**	1.13
USpowerGrid	1.85	**0.85**	2.18	0.67	**0.49**	1.37	0.37	**0.33**	1.12
commanche	10.78	**5.96**	1.81	3.63	**2.73**	1.33	1.51	**1.38**	1.09
LeHavre	12.75	**7.78**	1.64	4.90	**3.87**	1.27	2.19	**2.14**	1.02
pesa	9.61	**4.50**	2.14	4.14	**2.79**	1.48	1.50	**1.39**	1.08
bodyy5	12.53	**4.00**	3.13	4.31	**2.35**	1.83	2.01	**1.58**	1.27
finance256	10.44	**5.82**	1.79	4.27	**3.28**	1.30	2.28	**2.12**	1.08
btree	0.42	**0.24**	1.75	0.18	**0.14**	1.29	0.09	**0.08**	1.12
qh882	0.65	**0.38**	1.71	0.28	**0.21**	1.33	0.15	**0.14**	1.07
lpship04l	0.73	**0.55**	1.33	0.31	**0.26**	1.19	0.18	**0.17**	1.06

5 Conclusion

In this paper we proposed a sparse stress model that requires $\mathcal{O}(kn + m)$ space and time per iteration, and a preprocessing time of $\mathcal{O}(k(m + n \log n))$. While Barnes & Hut derive their representatives from a given partitioning, we argued that for our model it is more appropriate to first select the pivots and then to partition the graph only relying on its structure. Since the approximation quality heavily depends on the proper selection of these pivots, we evaluated different sampling techniques, showing that *k-means sp* works very well in practice. Additionally, we showed that using only a constant number of features for *k-means sp* in general does not reduce the quality of the resulting layout but decreases the overall running time.

Furthermore, we compared a variety of sparse stress models w.r.t. their performance in approximating the full stress model. We therefore proposed two new measures: the first one assesses the similarity of two layouts of the same graph via Gabriel graphs and the second one quantifies the visual error in a layout using convex hulls. For the tested graphs, all our experiments clearly showed that our proposed sparse stress model exceeds related approaches in approximating the full stress layout without compromising the computation time.

Table 6: Layouts and error charts of the algorithms. Each chart shows the zero y coordinate (black horizontal line), the median (red line), the 25 and 75 percentiles (black/gray ribbon) and the min/max error (outer black dashed line). The error (y-axis) is the difference between the Euclidean distance and the graph-theoretic distance (x-axis). 1000 bins have been used for weighted graphs.



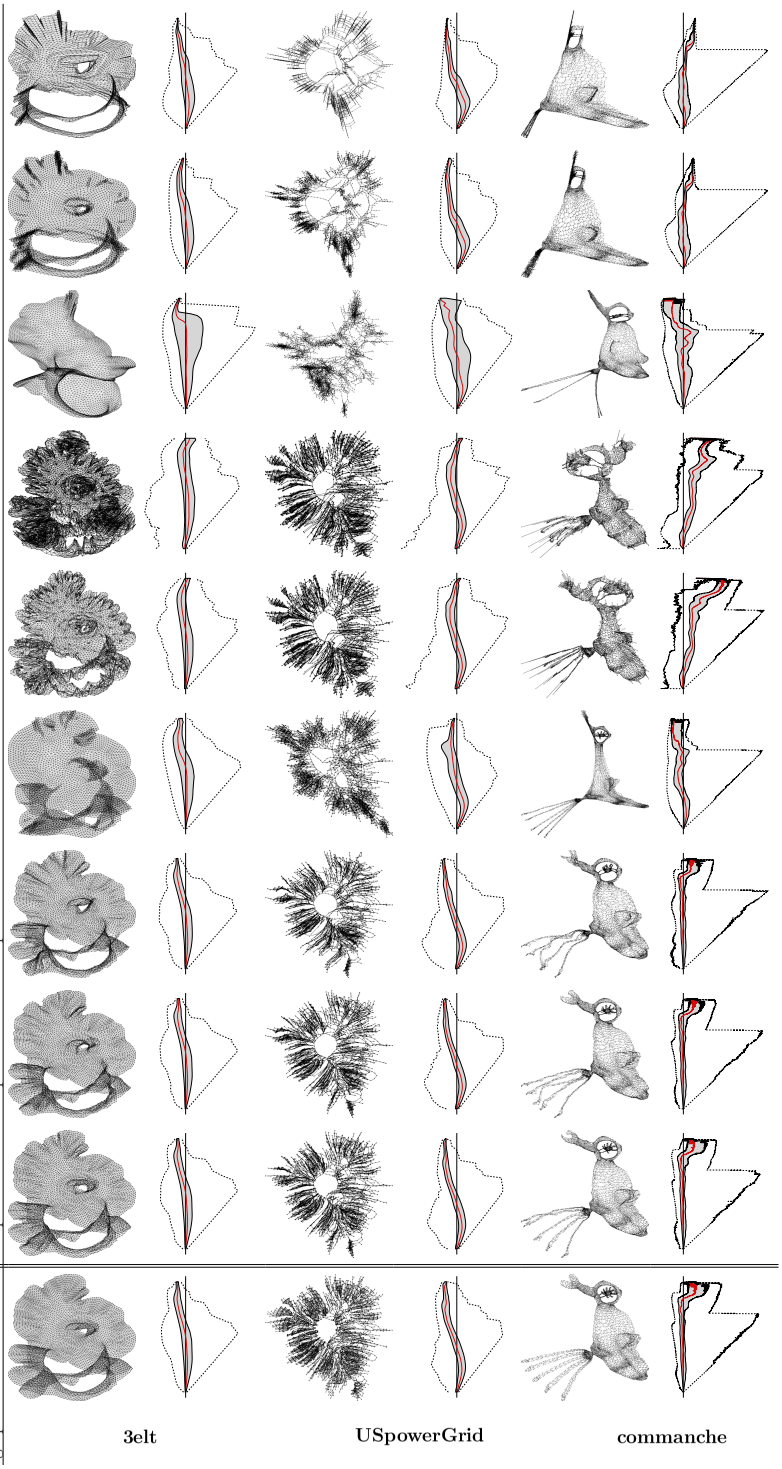


Table 6 (cont.): Layouts and error charts of the algorithms. Each chart shows the zero y coordinate (black horizontal line), the median (red line), the 25 and 75 percentiles (black/gray ribbon) and the min/max error (outer black dashed line). The error (y-axis) is the difference between the Euclidean distance and the graph-theoretic distance (x-axis). 1000 bins have been used for weighted graphs.

Table 6 (cont.): Layouts and error charts of the algorithms. Each chart shows the zero y coordinate (black horizontal line), the median (red line), the 25 and 75 percentiles (black/gray ribbon) and the min/max error (outer black dashed line). The error (y-axis) is the difference between the Euclidean distance and the graph-theoretic distance (x-axis). 1000 bins have been used for weighted graphs.

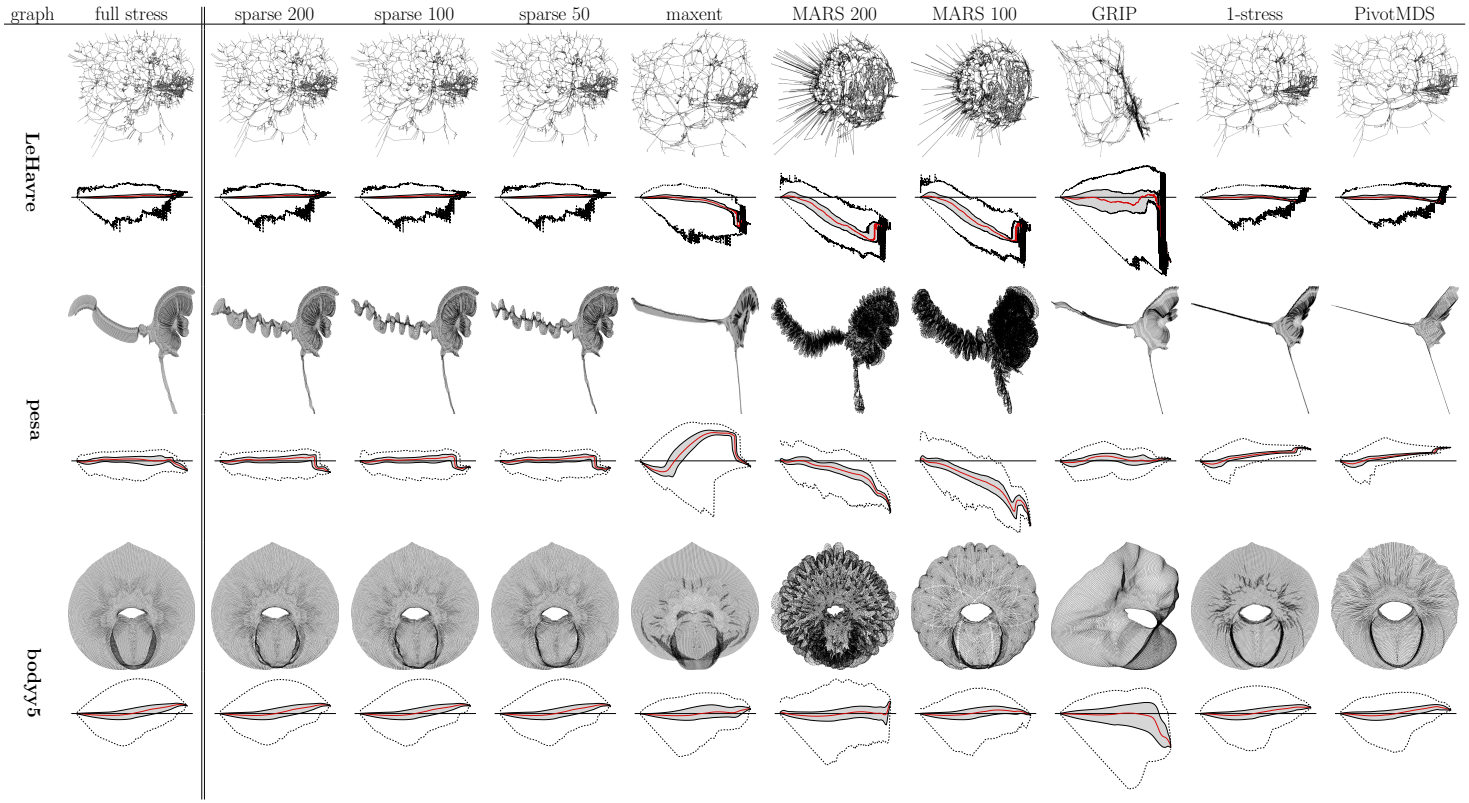


Table 6 (cont.): Layouts and error charts of the algorithms. Each chart shows the zero y coordinate (black horizontal line), the median (red line), the 25 and 75 percentiles (black/gray ribbon) and the min/max error (outer black dashed line). The error (y-axis) is the difference between the Euclidean distance and the graph-theoretic distance (x-axis). 1000 bins have been used for weighted graphs.

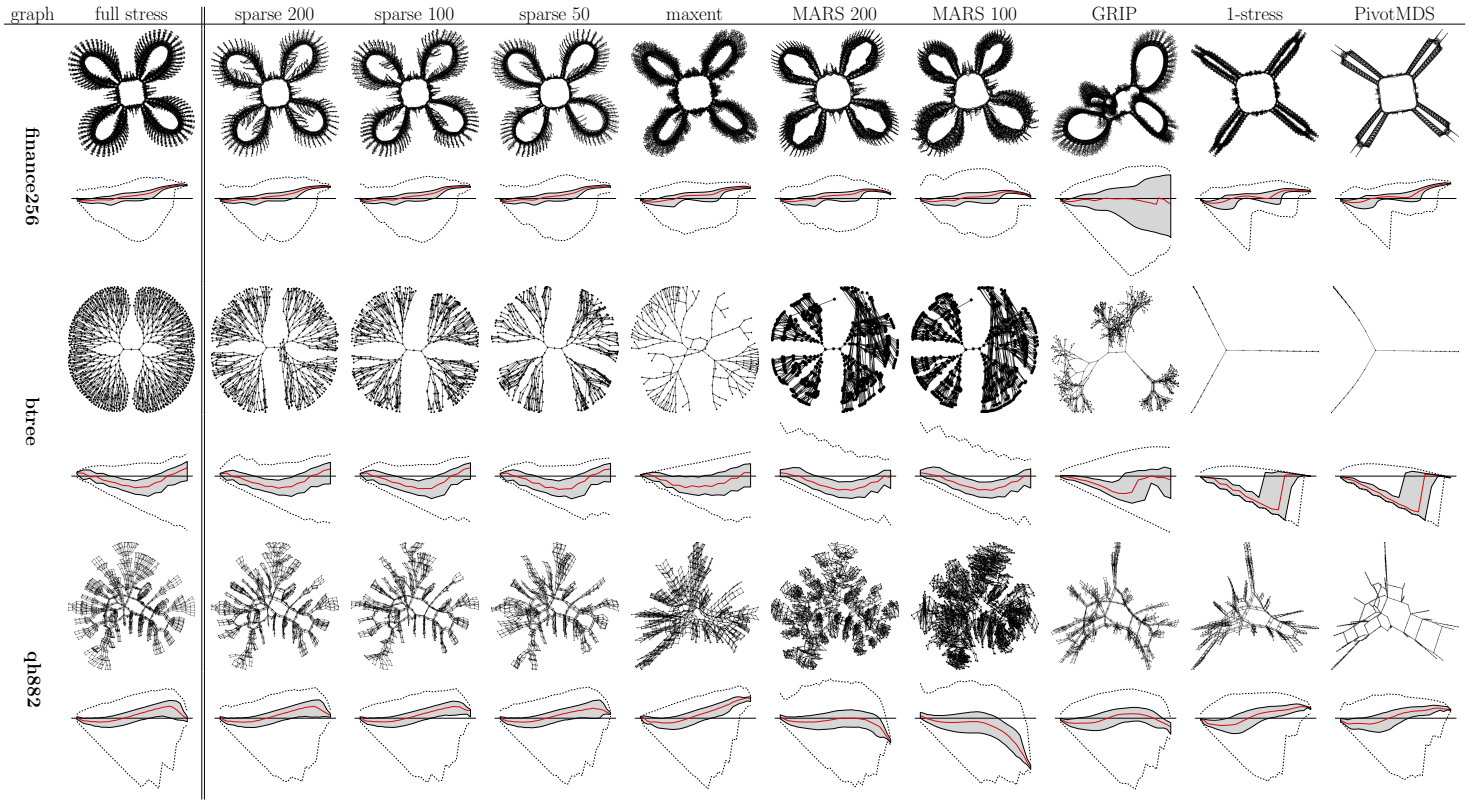
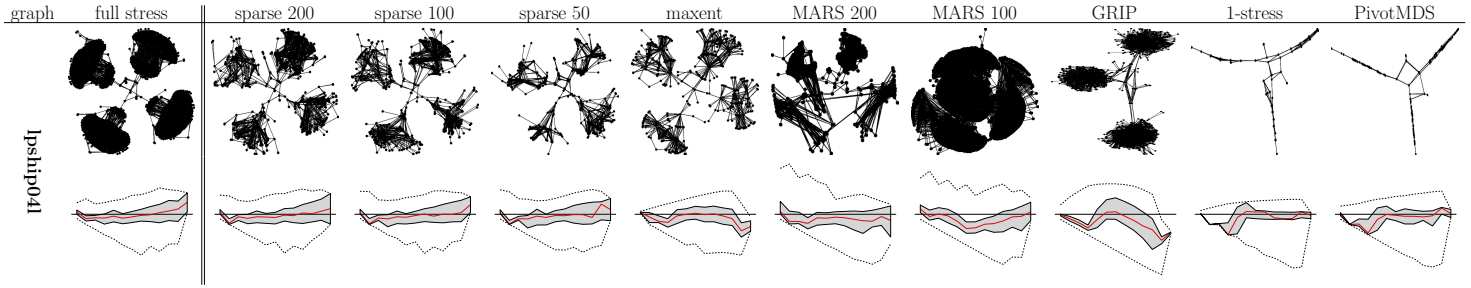


Table 6 (cont.): Layouts and error charts of the algorithms. Each chart shows the zero y coordinate (black horizontal line), the median (red line), the 25 and 75 percentiles (black/gray ribbon) and the min/max error (outer black dashed line). The error (y-axis) is the difference between the Euclidean distance and the graph-theoretic distance (x-axis). 1000 bins have been used for weighted graphs.



References

[1] J. Barnes and P. Hut. A hierarchical O($n \log n$) force-calculation algorithm. *Nature*, 324(6096):446–449, 1986. doi:10.1038/324446a0.

[2] I. Borg and P. J. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[3] U. Brandes. Drawing on physical analogies. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs: Methods and Models*, volume 2025 of *LNCS*, pages 71–86. Springer, 2001. doi:10.1007/3-540-44969-8_4.

[4] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In M. Kaufmann and D. Wagner, editors, *GD 2006*, volume 4372 of *LNCS*, pages 42–53. Springer, 2007. doi:10.1007/978-3-540-70904-6_6.

[5] U. Brandes and C. Pich. An experimental study on distance-based graph drawing. In I. G. Tollis and M. Patrignani, editors, *GD 2008*, volume 5417 of *LNCS*, pages 218–229. Springer, 2009. doi:10.1007/978-3-642-00219-9_21.

[6] U. Brandes, F. Schulz, D. Wagner, and T. Willhalm. Travel planning with self-made maps. In A. L. Buchsbaum and J. Snoeyink, editors, *ALENEX 2001*, volume 2153 of *LNCS*, pages 132–144. Springer, 2001. doi:10.1007/3-540-44808-X_10.

[7] J. D. Cohen. Drawing graphs to convey proximity: An incremental arrangement method. *ACM Transactions on Computer-Human Interaction*, 4(3):197–229, 1997. doi:10.1145/264645.264657.

[8] T. F. Cox and M. Cox. *Multidimensional Scaling, Second Edition*. Chapman and Hall/CRC, 2 edition, 2000.

[9] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1:1–1:25, 2011. doi:10.1145/2049662.2049663.

[10] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS 2002*, pages 705–712. MIT Press, 2002. URL: http://papers.nips.cc/paper/2141-global-versus-local-methods-in-nonlinear-dimensionality-reduction.

[11] P. Drineas, A. M. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56(1-3):9–33, 2004. doi:10.1023/B:MACH.0000033113.59016.96.

[12] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs. In R. Tamassia and I. G. Tollis, editors, *GD 1994*, volume 894 of *LNCS*, pages 388–403. Springer, 1995. doi:10.1007/3-540-58950-3_393.

[13] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi:10.1002/spe.4380211102.

[14] R. K. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969.

[15] P. Gajer, M. T. Goodrich, and S. G. Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. In J. Marks, editor, *GD 2000*, volume 1984 of *LNCS*, pages 211–221. Springer, 2001. doi:10.1007/3-540-44541-2_20.

[16] E. R. Gansner, Y. Hu, and S. Krishnan. COAST: A convex optimization approach to stress-based embedding. In S. K. Wismath and A. Wolff, editors, *GD 2013*, volume 8242 of *LNCS*, pages 268–279. Springer, 2013. doi:10.1007/978-3-319-03841-4_24.

[17] E. R. Gansner, Y. Hu, and S. C. North. A maxent-stress model for graph layout. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):927–940, 2013. doi:10.1109/TVCG.2012.299.

[18] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In J. Pach, editor, *GD 2004*, volume 3383 of *LNCS*, pages 239–250. Springer, 2004. doi:10.1007/978-3-540-31843-9_25.

[19] L. Greengard. *The Rapid evaluation of potential fields in particle systems*. ACM distinguished dissertations. Cambridge, Mass. MIT Press, 1988. URL: http://opac.inria.fr/record=b1086802.

[20] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In J. Pach, editor, *GD 2004*, volume 3383 of *LNCS*, pages 285–295. Springer, 2004. doi:10.1007/978-3-540-31843-9_29.

[21] K. M. Hall. An r-dimensional quadratic placement algorithm. *Managment Science*, 17(3):219–229, 1970. doi:10.1287/mnsc.17.3.219.

[22] Y. Hu and L. Shi. Visualizing large graphs. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(2):115–136, 2015. doi:10.1002/wics.1343.

[23] Y. F. Hu. Efficient and high quality force-directed graph drawing. *The Mathematica Journal*, 10:37–71, 2005. URL: http://www.mathematica-journal.com/issue/v10i1/contents/graph_draw/graph_draw.pdf.

[24] S. Ingram and T. Munzner. Glint: An MDS framework for costly distance functions. In A. Kerren and S. Seipel, editors, *SIGRAD 2012*, volume 81 of *Linköping Electronic Conference Proceedings*, pages 29–38. Linköping University Electronic Press, 2012. URL: http://www.ep.liu.se/ecp/article.asp?issue=081&article=005.

[25] M. Khoury, Y. Hu, S. Krishnan, and C. E. Scheidegger. Drawing large graphs by low-rank stress majorization. *Computer Graphics Forum*, 31(3):975–984, 2012. doi:10.1111/j.1467-8659.2012.03090.x.

[26] M. Klimenta and U. Brandes. Graph drawing by classical multidimensional scaling: New perspectives. In W. Didimo and M. Patrignani, editors, *GD 2012*, volume 7704 of *LNCS*, pages 55–66. Springer, 2013. doi:10.1007/978-3-642-36763-2_6.

[27] S. G. Kobourov. Force-directed drawing algorithms. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 383–408. Chapman and Hall/CRC, 2013.

[28] Y. Koren, L. Carmel, and D. Harel. ACE: A fast multiscale eigenvectors computation for drawing huge graphs. In P. C. Wong and K. Andrews, editors, *InfoVis 2002*, pages 137–144. IEEE Computer Society, 2002. doi:10.1109/INFVIS.2002.1173159.

[29] V. E. McGee. The multidimensional analysis of "elastic" distances. *British Journal of Mathematical and Statistical Psychology*, 19(2):181–196, 1966. doi:10.1111/j.2044-8317.1966.tb00367.x.

[30] H. Meyerhenke, M. Nöllenburg, and C. Schulz. Drawing large graphs by multilevel maxent-stress optimization. In E. D. Giacomo and A. Lubiw, editors, *GD 2015*, volume 9411 of *LNCS*, pages 30–43. Springer, 2015. doi:10.1007/978-3-319-27261-0_3.

[31] M. Ortmann, M. Klimenta, and U. Brandes. A sparse stress model. In Y. Hu and M. Nöllenburg, editors, *GD 2016*, volume 9801 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 2016. doi:10.1007/978-3-319-50106-2_2.

[32] A. J. Quigley. *Large Scale Relational Information Visualization, Clustering, and Abstraction.* PhD thesis, University of Newcastle, 2000.

[33] R. Sibson. Studies in the robustness of multidimensional scaling: Procrustes statistics. *Journal of the Royal Statistical Society. Series B (Methodological)*, 40(2):234–238, 1978. URL: http://www.jstor.org/stable/2984761.

[34] D. Tunkelang. JIGGLE: Java interactive graph layout environment. In S. Whitesides, editor, *GD 1998*, volume 1547 of *LNCS*, pages 412–422. Springer, 1998. doi:10.1007/3-540-37623-2_33.

[35] D. Tunkelang. *A Numerical Optimization Approach to General Graph Drawing*. PhD thesis, Carnegie Mellon University, 1999.

[36] C. Walshaw. A multilevel algorithm for force-directed graph-drawing. *Journal of Graph Algorithms and Applications*, 7(3):253–285, 2003. doi: 10.7155/jgaa.00070.