Drawing Big Graphs Using Spectral Sparsification

Peter Eades, Quan Nguyen(✉), and Seok-Hee Hong

The School of Information Technologies, University of Sydney, Sydney, Australia
{peter.eades,quan.nguyen,seokhee.hong}@sydney.edu.au

Abstract. Spectral sparsification is a general technique developed by Spielman *et al.* to reduce the number of edges in a graph while retaining its structural properties. We investigate the use of spectral sparsification to produce good visual representations of big graphs. We evaluate spectral sparsification approaches on real-world and synthetic graphs. We show that spectral sparsifiers are more effective than random edge sampling. Our results lead to guidelines for using spectral sparsification in big graph visualization.

1 Introduction

The problem of drawing very large graphs is challenging and has motivated a large body of research (see [15] for a survey). As the number of vertices and edges becomes larger, layout algorithms become less effective. Further, runtime is increased both at the layout stage and at the rendering stage. Recent work (for example [20]) approaches the problem by replacing the original graph with a "proxy graph". The proxy graph is typically much smaller than the original graph, and thus layout and rendering is easier. The challenge for the proxy graph approach is to ensure that the proxy graph is a good representation of the original graph; for visualization, we want the drawing of the proxy graph to be *faithful* [21] to the original graph.

In this paper we examine a specific proxy graph approach using *spectral sparsification* as introduced by Spielman *et al.* [1]: roughly speaking, the *spectrum* (that is, the eigenvalues of the Laplacian; see [10]) of the proxy graph approximates the spectrum of the original graph. Since the spectrum is closely related to graph-theoretic properties that are significant for graph drawing, this kind of proxy seems to promise faithful drawings.

We report results of an empirically investigation of the application of spectral sparsification to graph drawing. Specifically, we consider two closely related spectral sparsification techniques, one deterministic and one stochastic. We consider the quality of drawings so produced, using real-world and synthetic data sets. Quality is evaluated using the shape-based proxy graph metrics [20]. The results of spectral sparsification are compared to sparsifications obtained by simple random edge sampling. Our investigation confirms the promise of spectral sparsification, and shows that (overall) it is better than simple random edge sampling.

F. Frati and K.-L. Ma (Eds.): GD 2017, LNCS 10692, pp. 272–286, 2018.
https://doi.org/10.1007/978-3-319-73915-1_22

Section 2 recalls the proxy graph approach, and shape-based quality metrics for large graph drawing. Section 3 describes the application of spectral sparsification to graph visualization. Section 4 presents our experiments with spectral sparsification. The results of these experiments are presented and discussed in Sect. 5. Section 6 concludes.

2 Background

Proxy Graphs and Sparsification. The proxy graph approach is described in Fig. 1: for a given input graph G, a proxy graph G' and a drawing D' of G' are computed. The proxy graph represents G but is simpler and/or smaller than G in some sense. The user sees the drawing D' of G', and does not see a drawing of the original graph G. However, if G' is a "good" representation of G, then D' is an adequate visualization of G in that the user can see all the structure of G in the drawing D'.

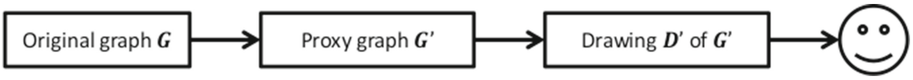


Fig. 1. In the *proxy graph approach*, the user sees a "proxy" of the original graph.

If G' is a subgraph of G, and the edge density of G' is smaller than the edge density of G, then we say that G' is a *sparsification* of G. Sparsification is the most common kind of proxy.

Sparsification has been extensively investigated in Graph Mining [9,16,19] (see survey [14]). Typically, sparsification is achieved by some kind of stochastic sampling. The most basic sparsification method is *random edge sampling* (RE): each edge is chosen independently with probability p [23]. This and many other simple stochastic strategies have been empirically investigated in the context of visualization of large graphs [20,29]. In this paper we apply a more sophisticated graph sparsification approach to visualization: the *spectral* sparsification work of Spielman *et al.* [1,25,26].

Shape-Based Quality Metrics. Traditional graph drawing metrics such as *edge bends*, *edge crossings*, and *angular resolution* are based on the *readability* of graphs; these metrics are good for small scale visualisation but become meaningless beyond a few hundred nodes [22]. For large graphs, *faithfulness metrics* are more important: informally, a drawing D of a graph G is *faithful* insofar as D determines G, that is, insofar as the mapping $G \rightarrow D$ is invertible.

Here we use *shape-based faithfulness metrics* [7]. The aim of these metrics is to measure how well the "shape" of the drawing represents the graph. For large graphs, such as in Fig. 2, the shape of the drawing is more significant than the number of edge bends and edge crossings. To make this notion more precise, we

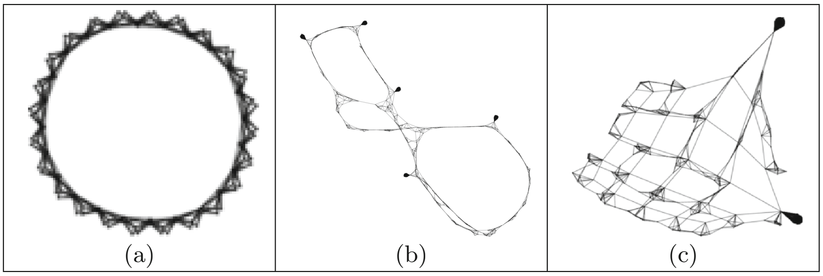


Fig. 2. The graphs *can_144*, *cN1031M22638*, and *gN733M62509*, drawn using FM^3. Note that each has a distinctive shape.

use "shape graphs". Given a set P of n points, a *shape graph* $S(P)$ is a graph with vertex set P such that the edge of $S(P)$ define the "shape" of P in some sense. Examples of shape graphs are the *Euclidean minimum spanning tree (EMST)*, the *relative neighbourhood graph (RNG)*, and the *Gabriel graph (GG)* [27].

Suppose that $G = (V, E)$ is a graph and P is a set of points in the plane, and each vertex $u \in V$ is associated with a point $p(u) \in P$. Denote the set of neighbours of u in G by $N_G(u)$, and the set of neighbours of $p(u)$ in the shape graph $S(P)$ by $N_{S(P)}(p(u))$. We say that

$$Jaccard(S(P), G) = \frac{1}{|V|} \sum_{u \in V} \frac{|N_G(u) \cap N_{S(P)}(p(u))|}{|N_G(u) \cup N_{S(P)}(p(u))|}$$

is the *Jaccard similarity* between the shape graph $S(P)$ and G. If D is a drawing of G then the *(shape-based) quality* of D is $Q(D, G) = Jaccard(P, G)$, where P is the set of vertex locations in the drawing D. Similarly, if D' is a drawing of a sparsification G' of G, then the *(shape-based) (proxy) quality* of D' is $Q(D', G) = Jaccard(P', G)$, where P' is the set of vertex locations in the drawing D'. Note that if u does not occur in D', we consider that $N'(u) = \emptyset$. For more details, see [8].

3 The Spectral Sparsification Approach to Large Graph Drawing

First we describe some of the terminology and concepts of spectral graph theory. More details are in standard texts; for example, [5,10][1]. The *adjacency matrix* of an n-vertex graph $G = (V, E)$ is the $n \times n$ matrix A, indexed by V, such that $A_{uv} = 1$ if $(u, v) \in E$ and $A_{uv} = 0$ otherwise. The *degree matrix* D of G is the diagonal matrix with where D_{uu} is the degree of vertex u. The *Laplacian* of G is $L = D - A$. The *spectrum* of G is the list $\lambda_1, \lambda_2, \ldots, \lambda_n$ of eigenvalues of L. It can be shown that L has real nonnegative eigenvalues [5], and we assume that $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$; straightforward computation shows that $\lambda_1 = 0$.

[1] Beware: much of the terminology in spectral graph theory is not standardised.

The spectrum of a graph is closely related to many structural properties of the graph:

Connectivity: The number of connected components of G is largest value of i for which $\lambda_i = 0$ [5]. Roughly speaking, a larger value of λ_2 indicates a more highly connected graph and is related to the diameter of the graph. If $\lambda_2 > 0$ then it is called the *algebraic connectivity* of G [11].

Clusters: *Spectral clustering* involves the projection of the graph using its smallest eigenvalues. Spectral clustering solves a relaxation of the *ratio cut* problem, that is, the problem of dividing a graph into clusters to minimise the ratio between the number of inter-cluster edges and the cluster size [28]. Informally, the ratio cut problem seeks to find clusters of similar size so that the coupling between clusters is minimised.

Stress: The spectrum solves a kind of constrained stress problem for the graph. More specifically, the Courant-Fischer theorem (see [5]) implies that

$$\lambda_i = \min_{x \in X_i} \sum_{(u,v) \in E} (x_u - x_v)^2, \tag{1}$$

where X_i is the set of unit vectors orthogonal to the first $(i-1)$ eigenvectors. The minimum is achieved when x is an eigenvalue corresponding to λ_i. Note that the right hand side of Eq. (1) is a kind of stress function.

Commute distance: The average time that a random walker takes to travel from vertex u to vertex v and return is the *commute distance* between u and v. Eigenvalues are related to random walks in the graph, and thus to commute distances (see [17]).

Spielman and Teng [26], following Benczur and Karger [3], first introduced the concept of "spectral approximation". Suppose that G is an n-vertex graph with Laplacian L, and G' is an n-vertex subgraph of G with Laplacian L'. If there is an $\epsilon > 0$ such that for every $x \in R^n$,

$$(1-\epsilon)\frac{x^T L' x}{x^T x} \le \frac{x^T L x}{x^T x} \le (1+\epsilon)\frac{x^T L' x}{x^T x}, \tag{2}$$

then G' is an *ϵ-spectral approximation* of G. Using the Courant-Fischer Theorem [5] with (2), one can show that if G' is an *ϵ-spectral approximation* of G then the eigenvalues and eigenvectors of G' are close to those of G. The importance of this is that spectral approximation preserves the structural properties listed above.

Spielman and Teng first showed that every n-vertex graph has a spectral approximation with $O(n \log n)$ edges [26]. The following theorem is one such result:

Theorem 1 ([26]). *Suppose that G is an n-vertex graph and $\frac{1}{\sqrt{n}} \le \epsilon \le 1$. Then with probability at least $\frac{1}{2}$, there is an ϵ-spectral approximation G' of G with $O(\frac{1}{\epsilon} n \log n)$ edges.*

Further research of Spielman *et al.* refines and improves spectral sparsification methods (see [1]). These results have potential for resolving scale issues in graph visualisation by reducing the size of the graph while retaining its (spectral) structure. However, the *practical impact* of these results for graph visualization is not clear, because of large constants involved.

The proof of Theorem 1 is essentially a stochastic sampling method, using the concept of "effective resistance". Suppose that we regard a graph $G = (V, E)$ as an electrical network where each edge is a $1 - \Omega$ resistor, and a current is applied. The voltage drop over an edge (u, v) is the *effective resistance* r_{uv} of (u, v). Effective resistance in a graph is closely related to commute distance, and can be computed simply from the Moore-Penrose inverse [2] of the Laplacian. If $L^\dagger$ is the Moore-Penrose inverse of L and $(u, v) \in E$, then $r_{uv} = L^\dagger_{uu} + L^\dagger_{vv} - 2L^\dagger_{uv}$.

We next describe two graph drawing algorithms, both variants of algorithms of Spielman *et al.* [1]. Each takes a graph G and an integer m', and computes a sparsification G' with m' edges, then draws G'.

`SSS` (Stochastic Spectral Sparsification) randomly selects edges with probability proportional to their resistance value. Let E' be the edge set from m' random selections. Let G' be the subgraph of G induced by E'; draw G'.

`DSS` (Deterministic Spectral Sparsification). Let E' consist of the m' of largest effective resistance. Let G' be the subgraph of G induced by E'; draw G'.

In both `DSS` and `SSS`, the sparsified graph can be drawn with any large-graph layout algorithm.

4 The Experiments

The driving hypothesis for this paper is that for large graphs, spectral sparsification gives good proxy graphs for visualization. To be more precise, we define the *relative density* of the sparsification G' for a graph G to be $\frac{m'}{m}$, where G has m edges and G' has m' edges. Note that a proxy with higher relative density should be a better approximation to the original graph; thus we expect that drawings of the proxy with higher relative density should have better quality.

Since spectral sparsification (approximately) preserves the eigenvalues, we hypothesize that both `SSS` and `DSS` are better than `RE`. Further, we expect that the difference becomes smaller when the relative density is larger. To state this precisely, let $D'_{\texttt{SSS}}$ (respectively $D'_{\texttt{RE}}$) denote the drawing obtained by `SSS` (respectively `RE`). We say that $\frac{Q(D'_{\texttt{SSS}}, G)}{Q(D'_{\texttt{RE}}, G)}$ is the *quality ratio* of `SSS`; similarly define the quality ratio of `DSS`. We expect that the quality ratio of both `SSS` and `DSS` is greater than 1. Further, we expect that the quality ratio for both algorithms tends to 1 as relative density tends to 1.

We implemented `DSS`, `SSS` and `RE` in Java, on top of the OpenIMAJ toolkit [13]. In particular, we used OpenIMAJ to compute the Moore-Penrose inverse. The experiments were performed on a Dell XPS 13 laptop, with an i7 Processor, 16 GB memory and 512 GB SSD. The laptop was running Ubuntu 16.04 with 20 GB swap memory. The computation of the Moore-Penrose inverse used Java 8, with a specified 16 GB heap. We used multiple threads to speed up the resistance computation.

We used three data sets. The first set of graphs is taken from "defacto-benchmark" graphs, including the Hachul library, Walshaw's Graph Partitioning Archive, the sparse matrices collection [6] and the network repository [24]. These include two types of graphs that have been extensively studied in graph drawing research: grid-like graphs and scale-free graphs. The second set is the GION data set [18]; this consists of RNA sequence graphs that are used for the analysis of repetitive sequences in sequencing data; these graphs have been used in previous experiments. They are locally dense and globally sparse, and generally have distinctive shapes. The third set consists of randomly generated graphs that contain interesting structures that are difficult to model with sparsification. Specifically, we generated a number of "black-hole graphs", each of which consists of one or more large and dense parts (so-called "black holes"), and these parts connect with the rest of the graph by relatively few edges. These relatively few edges *outside* the "black holes" determine the structure of the graph. Such graphs are difficult to sparsify because sampling strategies tend to take edges from the dense "black holes" and miss the structurally important edges. Tuesday, December 19, 2017 at 9:06 am Figures 2(b) and (c) are black-hole graphs. Details of the graphs that we used are in Table 1.

Table 1. Data sets

graph	$\|V\|$	$\|E\|$	type
can_144	144	576	grid
G_15	1785	20459	scalef
G_2	4970	7400	grid
G_3	2851	15093	grid
G_4	2075	4769	scalef
mm_0	3296	6432	grid
nasa1824	1824	18692	grid
facebook01	4039	88234	scalef
oflights	2939	15677	scalef
soc_h	2426	16630	scalef
yeastppi	2361	7182	scalef

(a) Benchmark graphs

graph	$\|V\|$	$\|E\|$
graph_1	5452	118404
graph_2	1159	6424
graph_3	7885	427406
graph_4	5953	186279
graph_5	1748	13957
graph_6	1785	20459
graph_7	3010	41757
graph_8	4924	52502

(b) GION graphs

graph	$\|V\|$	$\|E\|$
cN377M4790	377	4790
cN823M14995	823	14995
cN1031M226386	1031	22638
gN285M2009	285	2009
gN733M62509	733	62509
gN1080M17636	1080	17636
gN4784M38135	4784	38135

(c) Black-hole graphs

We sparsify these input graphs to a range of relative density values: from small (1%, 2%, 3%, 4%, 5%, 10%) to medium and large (15%, 20%, $\cdots$, 100%), using `SSS`, `DSS`, and `RE`.

For layout, we use the FM^3 algorithm [12], as implemented in OGDF [4]. However, we also confirmed our results using FM^3 variants (see Sect. 5.1).

We measured quality of the resulting visualizations by proxy quality metrics $Q(D, G)$ described in Sect. 2. For shape graphs, we used GG, RNG, and $EMST$;

the results for these three shape graphs are very similar, and here we report the results for *GG*.

5 Results from the Experiments

First we describe typical examples of the results of our experiments, using the graphs illustrated in Fig. 2; these are a relatively small defacto-benchmark graph *can_144*, and two black-hole graphs *cN1031M22638* and *gN733M62509*.

Sparsifications of *cN1031M22638* using RE, DSS, and SSS at relative densities of 3% and 15% are in Fig. 3. At relative density of 3%, both RE and SSS give poor results; the drawings do not show the structure of the graph. However, DSS gives a good representation. At relative density 15%, both DSS and SSS are good, while RE remains poor. A similar example, with relative densities of 1% and 10% for the black-hole graph *gN733M62509*, is in Fig. 4.

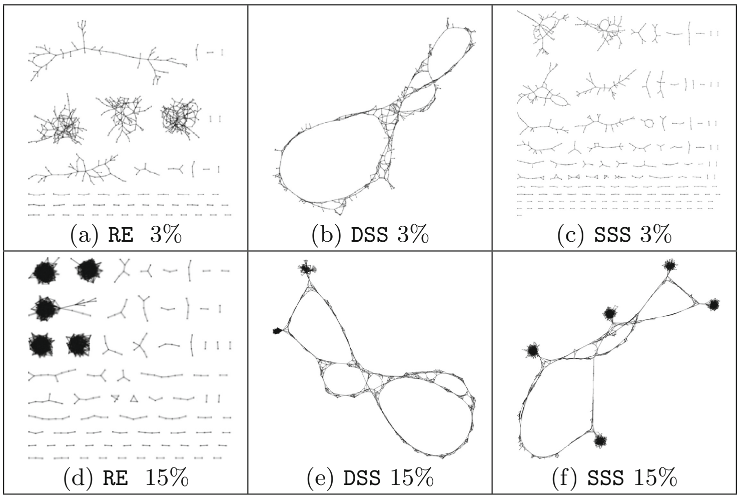


Fig. 3. Sparsifications of the graph cN1031M22638 at relative densities 3% and 15%.

While the results for *cN1031M22638* and *gN733M62509* are typical, some results did not fit this mold. For *can_144*, see Fig. 5; here RE and SSS give poor representations, even at very high relative density (40%). However, all three algorithms give good representations at relative density 50%.

5.1 Quality: Results and Observations

Figure 6 shows the quality metrics for the three data sets for all three algorithms. The x-axis shows relative densities from 1% to 95%; the y-axis shows quality measures of the proxies.

We make the following five observations from the results.

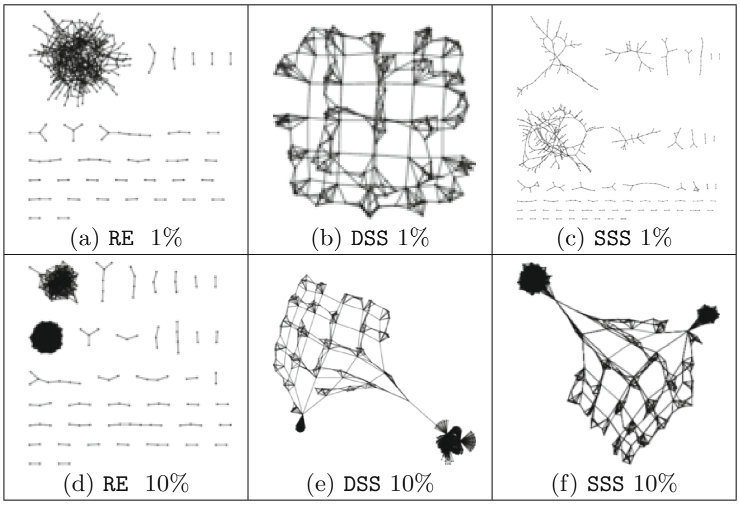


Fig. 4. Sparsifications of the graph *gN733M62509* at relative densities 1% and 10%.

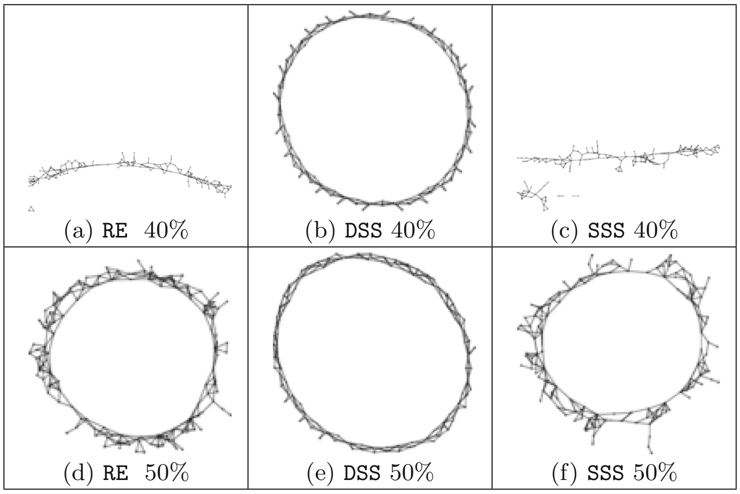
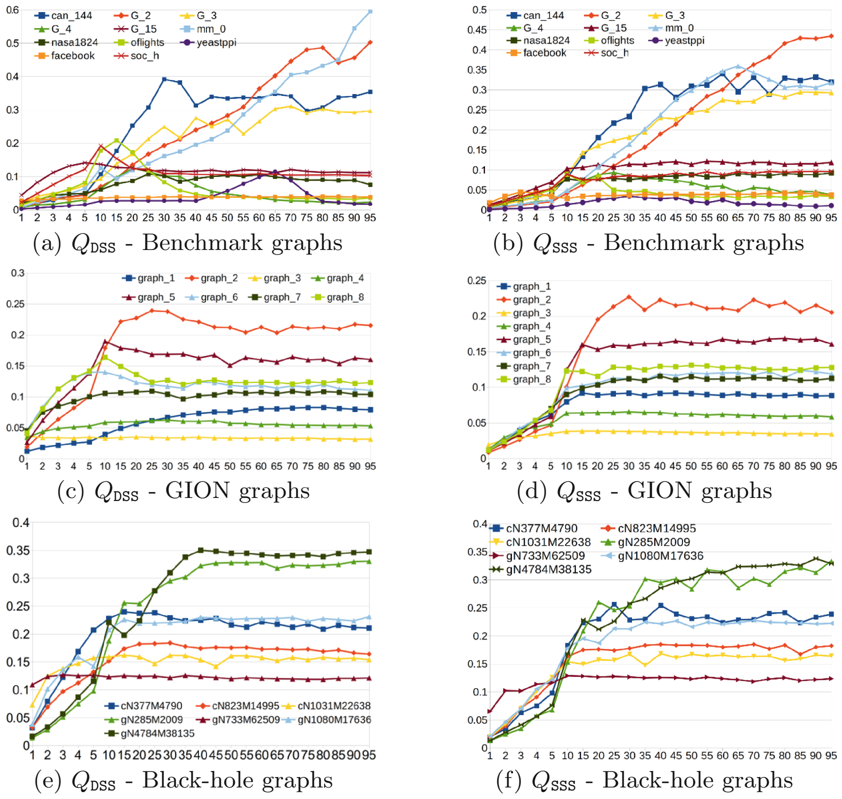


Fig. 5. Sparsifications of the graph *can_144* at relative densities 40% and 50%.

1. **Quality increases with relative density.** In general, quality increases as relative density increases. For many graphs there is a more interesting pattern: quality mostly increases up to a limit, achieved at a relative density between 10% and 30%, and then stays steady. Some of the defacto-benchmark graphs do not show this pattern: they show close to linear improvement in quality with density all the way up to 95%.
2. **Spectral sparsification is better than random edge sampling.** Figure 7 depicts the quality ratio (y-axis) for DSS and SSS for each data set, over

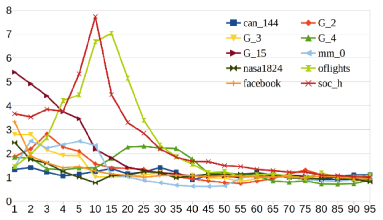


(a) $Q_{\texttt{DSS}}$ - Benchmark graphs
(b) $Q_{\texttt{SSS}}$ - Benchmark graphs
(c) $Q_{\texttt{DSS}}$ - GION graphs
(d) $Q_{\texttt{SSS}}$ - GION graphs
(e) $Q_{\texttt{DSS}}$ - Black-hole graphs
(f) $Q_{\texttt{SSS}}$ - Black-hole graphs

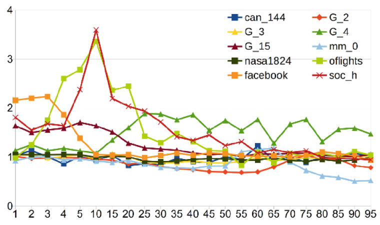
Fig. 6. Proxy quality metrics of output of `DSS`, `SSS` and `RE`: (a) and (b) defacto-benchmark graphs, (c) and (d) GION graphs, (e) and (f) black-hole graphs. The metrics use FM^3 layout and GG shape graphs. (Color figure online)

relative density from 1% to 95%. Note that the quality ratio is significant in most cases, especially at low relative density. For example, `DSS` metrics are around 200 times better than `RE`, and sometimes much more (for the yeast dataset it is about 400).
For most of the graphs, the quality ratio decreases as the relative density increases. Quality ratio is best for relative density smaller than 10%. When the relative density is more than 15%, `RE` may be slightly better than `DSS` for a few graphs, such as defacto-benchmark graphs *mm_0* graph (light blue), and *G_2* (red). Interestingly, *soc_h* and *oflights* show a peak at around 10% and 15% before a drop for larger relative density.

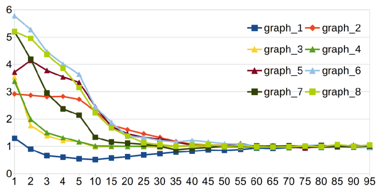
3. **Sparsification is better for grid-like graphs than for scale-free graphs.** Figure 8(a) shows the quality change for `DSS`, `SSS`, and `RE`with density, over the grid-like and scale-free defacto-benchmark graphs. Note that average values for `DSS` and `SSS` are better than the average value for `RE` when the relative density is less than 35%. When relative density is greater than



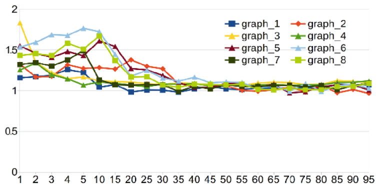
(a) $Q_{\texttt{DSS}}/Q_{\texttt{RE}}$ - Benchmark graphs



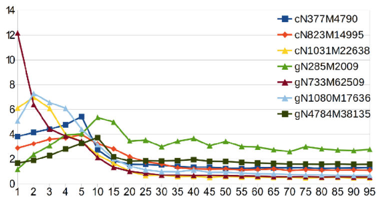
(b) $Q_{\texttt{SSS}}/Q_{\texttt{RE}}$ - Benchmark graphs



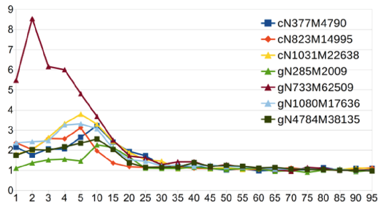
(c) $Q_{\texttt{DSS}}/Q_{\texttt{RE}}$ - GION graphs



(d) $Q_{\texttt{SSS}}/Q_{\texttt{RE}}$ - GION graphs



(e) $Q_{\texttt{DSS}}/Q_{\texttt{RE}}$ - Black-hole graphs



(f) $Q_{\texttt{SSS}}/Q_{\texttt{RE}}$ - Black-hole graphs

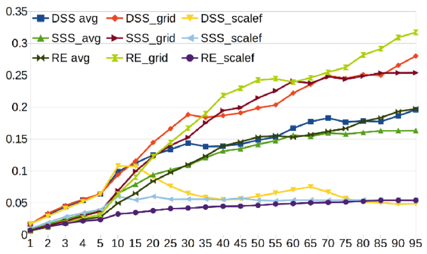
Fig. 7. Proxy quality ratio for `DSS` and `SSS`, for (a) and (b) defacto-benchmark graphs, (c) and (d) GION graphs, (e) and (f) black hole graphs. The y-axis shows the quality ratio. (Color figure online)

40%, there are fluctuations between `SSS` and `DSS`. For grid-like graphs, the `DSS` and `SSS` proxies give better average proxy measures than `RE` proxies for relative density less than 20%. For relative density greater than 35%, `RE` proxies improve. For scale-free graphs, `DSS` and `SSS` outperformed when relative density is under 80%.

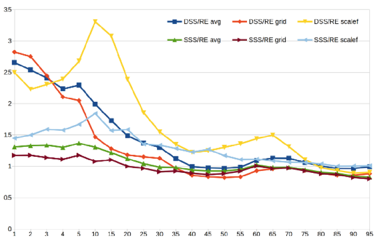
Figure 8(b) shows the ratio of the quality average between `DSS` over `RE` and `SSS` over `RE`. Overall, the quality ratios decline when relative density increase. The ratios are good from 1.2 to 3 times better for relative density up to 20%. For both types of graphs, `DSS` gives best quality, then `SSS` comes second.

4. **Deterministic spectral sparsification is better.** We compared the average of quality metrics for `DSS`, `SSS` and `RE` sparsification. Figure 9 shows the average quality values for the three data sets. As expected, average values increase when the relative density increases. Note that `DSS` gives the best average and `SSS` is the second best.

   Figure 10 shows the quality ratios $Q_{\texttt{DSS}}/Q_{\texttt{RE}}$ and $Q_{\texttt{SSS}}/Q_{\texttt{RE}}$ for all the data sets. Again, `DSS` gives an overall larger improvement over `RE` than `SSS`. The improvement of `DSS` over `RE` is good when relative density is less than 35%; `SSS` shows in improvement over `RE` as well, but it is not so dramatic.

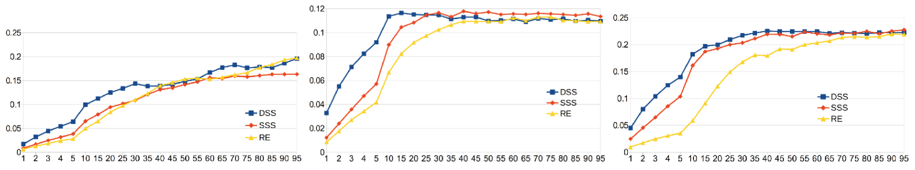


(a) Average quality metrics



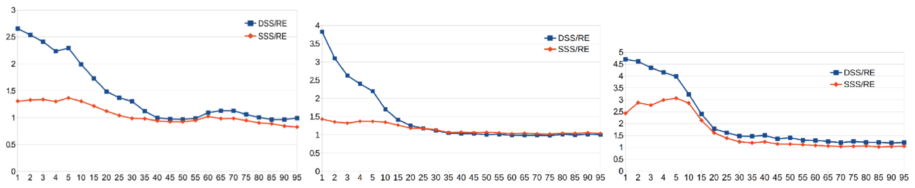
(b) Average quality ratio

Fig. 8. Comparison of proxy quality metrics of defacto-benchmark graphs: (1) Average quality measures, (2) Average of quality ratio. The values are computed by graph types for scale-free graphs (_scalef), grid-like (_grid) graphs, and overall (_avg). $Q_{\texttt{DSS}}/Q_{\texttt{RE}}$.



(a) Benchmark graphs (b) GION graphs (c) Black-hole graphs

Fig. 9. Average quality metrics of `DSS`, `SSS` and `RE` over all data sets.

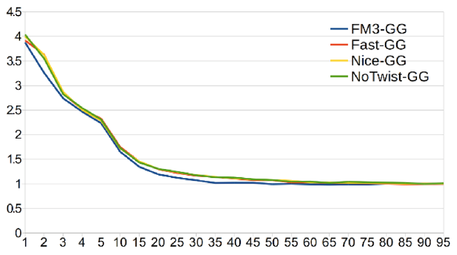


(a) Benchmark graphs (b) GION graphs (c) Black-hole graphs

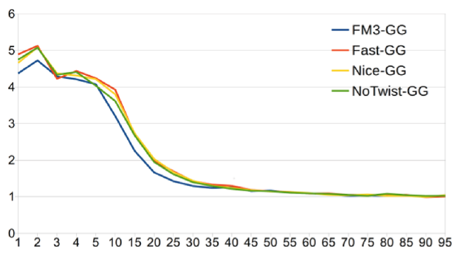
Fig. 10. Quality ratios of `DSS/RE` and `SSS/RE` over all data sets.

When relative density is beyond 35%, the ratio becomes small (close to 1) or even becomes smaller than 1. Further note from Fig. 10(a)–(c) that `DSS` and `SSS` give better quality ratios for black-hole graphs than for GION graphs and defacto-benchmark graphs.

5. **Quality results are consistent across different layout algorithms.** The results reported above use FM^3 for layout. However, we found that results using other layout algorithms were very similar. We measured the quality ratios using *FM*^{3}, *Fast*, *Nice* and *NoTwist* layouts from OGDF. For example, Fig. 11 shows the quality ratio of `DSS`. As depicted from the graphs, the improvement of `DSS` over `RE` is consistent across different layout algorithms. The differences in the ratios is very small.



(a) GION graphs



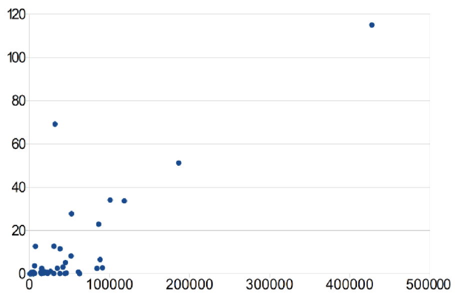
(b) Black-hole graphs

Fig. 11. Comparison of average quality ratio of DSS over RE between FM^3, Fast, Nice and NoTwist layouts. The y-axis shows the average quality ratio $Q_{\mathtt{DSS}}/Q_{\mathtt{RE}}$.

5.2 Runtime

Although the main purpose of our investigation was to evaluate the *effectiveness* of spectral sparsification, some remarks about runtime are in order.

Figure 12(a) illustrates runtimes. The x-axis shows the number of edges, and the y-axis shows the computation time in minutes. Figure 12(b) shows the amount of time for (parallel) computing resistance values. The x-axis shows the number of edges, and the y-axis shows the computation time in seconds.


(a) Inverse time

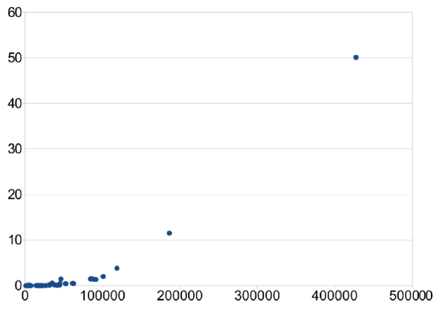

(b) Resistance computation time

Fig. 12. The running time of computing Moore-Penrose inverse (in minutes) and resistance values of all edges (in seconds).

The dominant part of runtime is the computation of the Moore-Penrose inverse (and thus effective resistance); for this we used standard software [13]. For the defacto-benchmark graphs, computing the Moore-Penrose inverse takes 10.42 min on average. Graph *can_144* takes minimum time for the Moore-Penrose inverse calculation (0.0003 min), and graph *graph_3* takes the longest time (115 min).

6 Concluding Remarks

This paper describes the first empirical study of the application of spectral sparsification in graph visualization.

Our experiments suggest that spectral sparsification approaches (DSS and SSS) are better than random edge approach. Further, the results suggest some guidelines for using spectral sparsification:

- DSS works better than SSS in practice.
- DSS and SSS give better quality metrics for grid-like graphs than for scale-free graphs.
- For sparsifications with low relative density (1% to 20%), DSS and SSS are considerably better than edge sampling. For relative density larger than 35%, RE may be more practical, because it is simpler, faster, and produces similar results to DSS and SSS.

Future work includes the following:

- Improve the runtime of these methods. For example, Spielman and Srivastava [25] present a nearly-linear time algorithm that builds a data structure from which we can query the approximate effective resistance between any two vertices in a graph in $O(\log n)$ time. This would allow testing spectral sparsification for larger graphs.
- More extensive evaluation: our experiments compare spectral sparsification with random edge sampling, but not with the wide range of sampling strategies above. Further, extension to larger data sets would be desirable.
- In our experiments, quality is measured using an objective shape-based metric. It would be useful to measure quality subjectively as well, using graph visualization experts as subjects in an HCI-style experiment.

References

1. Batson, J.D., Spielman, D.A., Srivastava, N., Teng, S.: Spectral sparsification of graphs: theory and algorithms. Commun. ACM **56**(8), 87–94 (2013)
2. Ben-Israel, A., Greville, T.N.: Generalized Inverses: Theory and Applications, vol. 15. Springer Science & Business Media, New York (2003)
3. Benczúr, A.A., Karger, D.R.: Approximating s-t minimum cuts in $\tilde{\mathrm{O}}(\mathrm{n}^2)$ time. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, 22–24 May 1996, pp. 47–55 (1996)
4. Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P.: The Open Graph Drawing Framework (OGDF). CRC Press, Boca Raton (2012)
5. Chung, F.: Spectral Graph Theory. American Maths Society (1997)
6. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Softw. **38**(1), 1:1–1:25 (2011)
7. Eades, P., Hong, S., Nguyen, A., Klein, K.: Shape-based quality metrics for large graph visualization. J. Graph Algorithms Appl. **21**(1), 29–53 (2017). https://doi.org/10.7155/jgaa.00405

8. Eades, P., Hong, S., Nguyen, A., Klein, K.: Shape-based quality metrics for large graph visualization. J. Graph Algorithms Appl. **21**(1), 29–53 (2017)
9. Gjoka, M., Kurant, M., Butts, C.T., Markopoulou, A.: Walking in facebook: a case study of unbiased sampling of OSNs. In: Proceedings of the 29th Conference on Information Communications, INFOCOM 2010 pp. 2498–2506. IEEE Press, Piscataway (2010)
10. Godsil, C.D., Royle, G.F.: Algebraic Graph Theory. Graduate Texts in Mathematics. Springer, New York (2001). https://doi.org/10.1007/978-1-4613-0163-9
11. Gross, J., Yellen, J.: Handbook of Graph Theory. CRC Press, Boca Raton (2004)
12. Hachul, S., Jünger, M.: Drawing large graphs with a potential-field-based multilevel algorithm. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 285–295. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_29
13. Hare, J.S., Samangooei, S., Dupplaw, D.: OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images. In: Proceedings of the 19th International Conference on Multimedia 2011, pp. 691–694 (2011)
14. Hu, P., Lau, W.C.: A survey and taxonomy of graph sampling. CoRR abs/1308.5865 (2013)
15. Hu, Y., Shi, L.: Visualizing large graphs. WIREs Comput. Stat. **7**(2), 115–136 (2015). https://doi.org/10.1002/wics.1343
16. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 631–636. ACM (2006)
17. Lovász, L.: Random walks on graphs: a survey. In: Miklós, D., Sós, V.T., Szőnyi, T. (eds.) Combinatorics, Paul Erdős is Eighty, vol. 2, pp. 353–398. János Bolyai Mathematical Society (1996)
18. Marner, M.R., Smith, R.T., Thomas, B.H., Klein, K., Eades, P., Hong, S.-H.: GION: interactively untangling large graphs on wall-sized displays. In: Duncan, C., Symvonis, A. (eds.) GD 2014. LNCS, vol. 8871, pp. 113–124. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45803-7_10
19. Morstatter, F., Pfeffer, J., Liu, H., Carley, K.: Is the sample good enough? Comparing data from Twitter's streaming API with Twitter's firehose, pp. 400–408. AAAI press (2013)
20. Nguyen, Q.H., Hong, S.H., Eades, P., Meidiana, A.: Proxy graph: visual quality metrics of big graph sampling. IEEE Trans. Visual Comput. Graphics **23**(6), 1600–1611 (2017)
21. Nguyen, Q., Eades, P., Hong, S.-H.: On the faithfulness of graph visualizations. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 566–568. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36763-2_55
22. Nguyen, Q.H., Eades, P., Hong, S.: On the faithfulness of graph visualizations. In: IEEE Pacific Visualization Symposium, PacificVis 2013, 27 February–1 March 2013, Sydney, NSW, Australia, pp. 209–216 (2013). https://doi.org/10.1109/PacificVis.2013.6596147
23. Rafiei, D., Curial, S.: Effectively visualizing large networks through sampling. In: 16th IEEE Visualization Conference, VIS 2005, Minneapolis, MN, USA, 23–28 October 2005, pp. 375–382 (2005)
24. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). http://networkrepository.com
25. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. CoRR abs/0803.0929 (2008). http://arxiv.org/abs/0803.0929

26. Spielman, D.A., Teng, S.: Spectral sparsification of graphs. SIAM J. Comput. **40**(4), 981–1025 (2011)
27. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. Pattern Recogn. **12**(4), 261–268 (1980). https://doi.org/10.1016/0031-3203(80)90066-7
28. Von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
29. Wu, Y., Cao, N., Archambault, D., Shen, Q., Qu, H., Cui, W.: Evaluation of graph sampling: a visualization perspective. IEEE Trans. Visual Comput. Graphics **23**(1), 401–410 (2017)